



minnesota educational computing consortium

*Beginning
Applesoft
BASIC*

Training Materials

For Use with the APPLE® II Computer

#648

***Beginning
Applesoft
BASIC***

Training Materials

For Use with the APPLE® II Computer

© Minnesota Educational Computing Consortium
3490 Lexington Avenue North
St. Paul, MN 55112

September 1, 1983

APPLE IITM and ApplesoftTM are registered trademarks of Apple Computer, Inc. of Cupertino, California. The Apple II, Apple II+, and Apple //e are referenced hereafter in this manual as the Apple. Permission is granted to duplicate classroom sets of the diskette and student materials contained in this manual.

TABLE OF CONTENTS

Introduction	1
Instructor Notes	7
Session I: Printing and Drawing - Output	8
Session II: Variables	14
Session III: Repeating	18
Session IV: Making Decisions	22
Session V: Accumulators, Counters, and Flags	27
Masters for Transparencies	31
Masters for Activity Handouts	37
Reference Materials	63
(Pages within this section are numbered R-1 through R-14)	
MECC Services	

TABLE OF CONTENTS

1	Introduction
2	Chapter I
3	Chapter II
4	Chapter III
5	Chapter IV
6	Chapter V
7	Chapter VI
8	Chapter VII
9	Chapter VIII
10	Chapter IX
11	Chapter X
12	Chapter XI
13	Chapter XII
14	Chapter XIII
15	Chapter XIV
16	Chapter XV
17	Chapter XVI
18	Chapter XVII
19	Chapter XVIII
20	Chapter XIX
21	Chapter XX
22	Chapter XXI
23	Chapter XXII
24	Chapter XXIII
25	Chapter XXIV
26	Chapter XXV
27	Chapter XXVI
28	Chapter XXVII
29	Chapter XXVIII
30	Chapter XXIX
31	Chapter XXX
32	Chapter XXXI
33	Chapter XXXII
34	Chapter XXXIII
35	Chapter XXXIV
36	Chapter XXXV
37	Chapter XXXVI
38	Chapter XXXVII
39	Chapter XXXVIII
40	Chapter XXXIX
41	Chapter XL
42	Chapter XLI
43	Chapter XLII
44	Chapter XLIII
45	Chapter XLIV
46	Chapter XLV
47	Chapter XLVI
48	Chapter XLVII
49	Chapter XLVIII
50	Chapter XLIX
51	Chapter L
52	Chapter LI
53	Chapter LII
54	Chapter LIII
55	Chapter LIV
56	Chapter LV
57	Chapter LVI
58	Chapter LVII
59	Chapter LVIII
60	Chapter LIX
61	Chapter LX
62	Chapter LXI
63	Chapter LXII
64	Chapter LXIII
65	Chapter LXIV
66	Chapter LXV
67	Chapter LXVI
68	Chapter LXVII
69	Chapter LXVIII
70	Chapter LXIX
71	Chapter LXX
72	Chapter LXXI
73	Chapter LXXII
74	Chapter LXXIII
75	Chapter LXXIV
76	Chapter LXXV
77	Chapter LXXVI
78	Chapter LXXVII
79	Chapter LXXVIII
80	Chapter LXXIX
81	Chapter LXXX
82	Chapter LXXXI
83	Chapter LXXXII
84	Chapter LXXXIII
85	Chapter LXXXIV
86	Chapter LXXXV
87	Chapter LXXXVI
88	Chapter LXXXVII
89	Chapter LXXXVIII
90	Chapter LXXXIX
91	Chapter LXXXX
92	Chapter LXXXXI
93	Chapter LXXXXII
94	Chapter LXXXXIII
95	Chapter LXXXXIV
96	Chapter LXXXXV
97	Chapter LXXXXVI
98	Chapter LXXXXVII
99	Chapter LXXXXVIII
100	Chapter LXXXXIX
101	Chapter LXXXXX
102	Chapter LXXXXXI
103	Chapter LXXXXXII
104	Chapter LXXXXXIII
105	Chapter LXXXXXIV
106	Chapter LXXXXXV
107	Chapter LXXXXXVI
108	Chapter LXXXXXVII
109	Chapter LXXXXXVIII
110	Chapter LXXXXXIX
111	Chapter LXXXXXX
112	Chapter LXXXXXXI
113	Chapter LXXXXXXII
114	Chapter LXXXXXXIII
115	Chapter LXXXXXXIV
116	Chapter LXXXXXXV
117	Chapter LXXXXXXVI
118	Chapter LXXXXXXVII
119	Chapter LXXXXXXVIII
120	Chapter LXXXXXXIX
121	Chapter LXXXXXXX
122	Chapter LXXXXXXXI
123	Chapter LXXXXXXXII
124	Chapter LXXXXXXXIII
125	Chapter LXXXXXXXIV
126	Chapter LXXXXXXXV
127	Chapter LXXXXXXXVI
128	Chapter LXXXXXXXVII
129	Chapter LXXXXXXXVIII
130	Chapter LXXXXXXXIX
131	Chapter LXXXXXXXI
132	Chapter LXXXXXXXII
133	Chapter LXXXXXXXIII
134	Chapter LXXXXXXXIV
135	Chapter LXXXXXXXV
136	Chapter LXXXXXXXVI
137	Chapter LXXXXXXXVII
138	Chapter LXXXXXXXVIII
139	Chapter LXXXXXXXIX
140	Chapter LXXXXXXXI
141	Chapter LXXXXXXXII
142	Chapter LXXXXXXXIII
143	Chapter LXXXXXXXIV
144	Chapter LXXXXXXXV
145	Chapter LXXXXXXXVI
146	Chapter LXXXXXXXVII
147	Chapter LXXXXXXXVIII
148	Chapter LXXXXXXXIX
149	Chapter LXXXXXXXI
150	Chapter LXXXXXXXII
151	Chapter LXXXXXXXIII
152	Chapter LXXXXXXXIV
153	Chapter LXXXXXXXV
154	Chapter LXXXXXXXVI
155	Chapter LXXXXXXXVII
156	Chapter LXXXXXXXVIII
157	Chapter LXXXXXXXIX
158	Chapter LXXXXXXXI
159	Chapter LXXXXXXXII
160	Chapter LXXXXXXXIII
161	Chapter LXXXXXXXIV
162	Chapter LXXXXXXXV
163	Chapter LXXXXXXXVI
164	Chapter LXXXXXXXVII
165	Chapter LXXXXXXXVIII
166	Chapter LXXXXXXXIX
167	Chapter LXXXXXXXI
168	Chapter LXXXXXXXII
169	Chapter LXXXXXXXIII
170	Chapter LXXXXXXXIV
171	Chapter LXXXXXXXV
172	Chapter LXXXXXXXVI
173	Chapter LXXXXXXXVII
174	Chapter LXXXXXXXVIII
175	Chapter LXXXXXXXIX
176	Chapter LXXXXXXXI
177	Chapter LXXXXXXXII
178	Chapter LXXXXXXXIII
179	Chapter LXXXXXXXIV
180	Chapter LXXXXXXXV
181	Chapter LXXXXXXXVI
182	Chapter LXXXXXXXVII
183	Chapter LXXXXXXXVIII
184	Chapter LXXXXXXXIX
185	Chapter LXXXXXXXI
186	Chapter LXXXXXXXII
187	Chapter LXXXXXXXIII
188	Chapter LXXXXXXXIV
189	Chapter LXXXXXXXV
190	Chapter LXXXXXXXVI
191	Chapter LXXXXXXXVII
192	Chapter LXXXXXXXVIII
193	Chapter LXXXXXXXIX
194	Chapter LXXXXXXXI
195	Chapter LXXXXXXXII
196	Chapter LXXXXXXXIII
197	Chapter LXXXXXXXIV
198	Chapter LXXXXXXXV
199	Chapter LXXXXXXXVI
200	Chapter LXXXXXXXVII
201	Chapter LXXXXXXXVIII
202	Chapter LXXXXXXXIX
203	Chapter LXXXXXXXI
204	Chapter LXXXXXXXII
205	Chapter LXXXXXXXIII
206	Chapter LXXXXXXXIV
207	Chapter LXXXXXXXV
208	Chapter LXXXXXXXVI
209	Chapter LXXXXXXXVII
210	Chapter LXXXXXXXVIII
211	Chapter LXXXXXXXIX
212	Chapter LXXXXXXXI
213	Chapter LXXXXXXXII
214	Chapter LXXXXXXXIII
215	Chapter LXXXXXXXIV
216	Chapter LXXXXXXXV
217	Chapter LXXXXXXXVI
218	Chapter LXXXXXXXVII
219	Chapter LXXXXXXXVIII
220	Chapter LXXXXXXXIX
221	Chapter LXXXXXXXI
222	Chapter LXXXXXXXII
223	Chapter LXXXXXXXIII
224	Chapter LXXXXXXXIV
225	Chapter LXXXXXXXV
226	Chapter LXXXXXXXVI
227	Chapter LXXXXXXXVII
228	Chapter LXXXXXXXVIII
229	Chapter LXXXXXXXIX
230	Chapter LXXXXXXXI
231	Chapter LXXXXXXXII
232	Chapter LXXXXXXXIII
233	Chapter LXXXXXXXIV
234	Chapter LXXXXXXXV
235	Chapter LXXXXXXXVI
236	Chapter LXXXXXXXVII
237	Chapter LXXXXXXXVIII
238	Chapter LXXXXXXXIX
239	Chapter LXXXXXXXI
240	Chapter LXXXXXXXII
241	Chapter LXXXXXXXIII
242	Chapter LXXXXXXXIV
243	Chapter LXXXXXXXV
244	Chapter LXXXXXXXVI
245	Chapter LXXXXXXXVII
246	Chapter LXXXXXXXVIII
247	Chapter LXXXXXXXIX
248	Chapter LXXXXXXXI
249	Chapter LXXXXXXXII
250	Chapter LXXXXXXXIII
251	Chapter LXXXXXXXIV
252	Chapter LXXXXXXXV
253	Chapter LXXXXXXXVI
254	Chapter LXXXXXXXVII
255	Chapter LXXXXXXXVIII
256	Chapter LXXXXXXXIX
257	Chapter LXXXXXXXI
258	Chapter LXXXXXXXII
259	Chapter LXXXXXXXIII
260	Chapter LXXXXXXXIV
261	Chapter LXXXXXXXV
262	Chapter LXXXXXXXVI
263	Chapter LXXXXXXXVII
264	Chapter LXXXXXXXVIII
265	Chapter LXXXXXXXIX
266	Chapter LXXXXXXXI
267	Chapter LXXXXXXXII
268	Chapter LXXXXXXXIII
269	Chapter LXXXXXXXIV
270	Chapter LXXXXXXXV
271	Chapter LXXXXXXXVI
272	Chapter LXXXXXXXVII
273	Chapter LXXXXXXXVIII
274	Chapter LXXXXXXXIX
275	Chapter LXXXXXXXI
276	Chapter LXXXXXXXII
277	Chapter LXXXXXXXIII
278	Chapter LXXXXXXXIV
279	Chapter LXXXXXXXV
280	Chapter LXXXXXXXVI
281	Chapter LXXXXXXXVII
282	Chapter LXXXXXXXVIII
283	Chapter LXXXXXXXIX
284	Chapter LXXXXXXXI
285	Chapter LXXXXXXXII
286	Chapter LXXXXXXXIII
287	Chapter LXXXXXXXIV
288	Chapter LXXXXXXXV
289	Chapter LXXXXXXXVI
290	Chapter LXXXXXXXVII
291	Chapter LXXXXXXXVIII
292	Chapter LXXXXXXXIX
293	Chapter LXXXXXXXI
294	Chapter LXXXXXXXII
295	Chapter LXXXXXXXIII
296	Chapter LXXXXXXXIV
297	Chapter LXXXXXXXV
298	Chapter LXXXXXXXVI
299	Chapter LXXXXXXXVII
300	Chapter LXXXXXXXVIII
301	Chapter LXXXXXXXIX
302	Chapter LXXXXXXXI
303	Chapter LXXXXXXXII
304	Chapter LXXXXXXXIII
305	Chapter LXXXXXXXIV
306	Chapter LXXXXXXXV
307	Chapter LXXXXXXXVI
308	Chapter LXXXXXXXVII
309	Chapter LXXXXXXXVIII
310	Chapter LXXXXXXXIX
311	Chapter LXXXXXXXI
312	Chapter LXXXXXXXII
313	Chapter LXXXXXXXIII
314	Chapter LXXXXXXXIV
315	Chapter LXXXXXXXV
316	Chapter LXXXXXXXVI
317	Chapter LXXXXXXXVII
318	Chapter LXXXXXXXVIII
319	Chapter LXXXXXXXIX
320	Chapter LXXXXXXXI
321	Chapter LXXXXXXXII
322	Chapter LXXXXXXXIII
323	Chapter LXXXXXXXIV
324	Chapter LXXXXXXXV
325	Chapter LXXXXXXXVI
326	Chapter LXXXXXXXVII
327	Chapter LXXXXXXXVIII
328	Chapter LXXXXXXXIX
329	Chapter LXXXXXXXI
330	Chapter LXXXXXXXII
331	Chapter LXXXXXXXIII
332	Chapter LXXXXXXXIV
333	Chapter LXXXXXXXV
334	Chapter LXXXXXXXVI
335	Chapter LXXXXXXXVII
336	Chapter LXXXXXXXVIII
337	Chapter LXXXXXXXIX
338	Chapter LXXXXXXXI
339	Chapter LXXXXXXXII
340	Chapter LXXXXXXXIII
341	Chapter LXXXXXXXIV
342	Chapter LXXXXXXXV
343	Chapter LXXXXXXXVI
344	Chapter LXXXXXXXVII
345	Chapter LXXXXXXXVIII
346	Chapter LXXXXXXXIX
347	Chapter LXXXXXXXI
348	Chapter LXXXXXXXII
349	Chapter LXXXXXXXIII
350	Chapter LXXXXXXXIV
351	Chapter LXXXXXXXV
352	Chapter LXXXXXXXVI
353	Chapter LXXXXXXXVII
354	Chapter LXXXXXXXVIII
355	Chapter LXXXXXXXIX
356	Chapter LXXXXXXXI
357	Chapter LXXXXXXXII
358	Chapter LXXXXXXXIII
359	Chapter LXXXXXXXIV
360	Chapter LXXXXXXXV
361	Chapter LXXXXXXXVI
362	Chapter LXXXXXXXVII
363	Chapter LXXXXXXXVIII
364	Chapter LXXXXXXXIX
365	Chapter LXXXXXXXI
366	Chapter LXXXXXXXII
367	Chapter LXXXXXXXIII
368	Chapter LXXXXXXXIV
369	Chapter LXXXXXXXV
370	Chapter LXXXXXXXVI
371	Chapter LXXXXXXXVII
372	Chapter LXXXXXXXVIII
373	Chapter LXXXXXXXIX
374	Chapter LXXXXXXXI
375	Chapter LXXXXXXXII
376	Chapter LXXXXXXXIII
377	Chapter LXXXXXXXIV
378	Chapter LXXXXXXXV
379	Chapter LXXXXXXXVI
380	Chapter LXXXXXXXVII
381	Chapter LXXXXXXXVIII
382	Chapter LXXXXXXXIX
383	Chapter LXXXXXXXI
384	Chapter LXXXXXXXII
385	Chapter LXXXXXXXIII
386	Chapter LXXXXXXXIV
387	Chapter LXXXXXXXV
388	Chapter LXXXXXXXVI
389	Chapter LXXXXXXXVII
390	Chapter LXXXXXXXVIII
391	Chapter LXXXXXXXIX
392	Chapter LXXXXXXXI
393	Chapter LXXXXXXXII
394	Chapter LXXXXXXXIII
395	Chapter LXXXXXXXIV
396	Chapter LXXXXXXXV
397	Chapter LXXXXXXXVI
398	Chapter LXXXXXXXVII
399	Chapter LXXXXXXXVIII
400	Chapter LXXXXXXXIX
401	Chapter LXXXXXXXI
402	Chapter LXXXXXXXII
403	Chapter LXXXXXXXIII
404	Chapter LXXXXXXXIV
405	Chapter LXXXXXXXV
406	Chapter LXXXXXXXVI
407	Chapter LXXXXXXXVII
408	Chapter LXXXXXXXVIII
409	Chapter LXXXXXXXIX
410	Chapter LXXXXXXXI
411	Chapter LXXXXXXXII
412	Chapter LXXXXXXXIII
413	Chapter LXXXXXXXIV
414	Chapter LXXXXXXXV
415	Chapter LXXXXXXXVI
416	Chapter LXXXXXXXVII
417	Chapter LXXXXXXXVIII
418	Chapter LXXXXXXXIX
419	Chapter LXXXXXXXI
420	Chapter LXXXXXXXII
421	Chapter LXXXXXXXIII
422	Chapter LXXXXXXXIV
423	Chapter LXXXXXXXV
424	Chapter LXXXXXXXVI
425	Chapter LXXXXXXXVII
426	Chapter LXXXXXXXVIII
427	Chapter LXXXXXXXIX
428	Chapter LXXXXXXXI
429	Chapter LXXXXXXXII
430	Chapter LXXXXXXXIII
431	Chapter LXXXXXXXIV
432	Chapter LXXXXXXXV
433	Chapter LXXXXXXXVI
434	Chapter LXXXXXXXVII
435	Chapter LXXXXXXXVIII
436	Chapter LXXXXXXXIX
437	Chapter LXXXXXXXI
438	Chapter LXXXXXXXII
439	Chapter LXXXXXXXIII
440	Chapter LXXXXXXXIV
441	Chapter LXXXXXXXV
442	Chapter LXXXXXXXVI
443	Chapter LXXXXXXXVII
444	Chapter LXXXXXXXVIII
445	Chapter LXXXXXXXIX
446	Chapter LXXXXXXXI
447	Chapter LXXXXXXXII
448	Chapter LXXXXXXXIII
449	Chapter LXXXXXXXIV
450	Chapter LXXXXXXXV
451	Chapter LXXXXXXXVI
452	Chapter LXXXXXXXVII
453	Chapter LXXXXXXXVIII
454	Chapter LXXXXXXXIX
455	Chapter LXXXXXXXI
456	Chapter LXXXXXXXII
457	Chapter LXXXXXXXIII
458	Chapter LXXXXXXXIV
459	Chapter LXXXXXXXV
460	Chapter LXXXXXXXVI
461	Chapter LXXXXXXXVII
462	Chapter LXXXXXXXVIII
463	Chapter LXXXXXXXIX
464	Chapter LXXXXXXXI
465	Chapter LXXXXXXXII
466	Chapter LXXXXXXXIII
467	Chapter LXXXXXXXIV
468	Chapter LXXXXXXXV
469	Chapter LXXXXXXXVI
470	Chapter LXXXXXXXVII
471	Chapter LXXXXXXXVIII
472	Chapter LXXXXXXXIX
473	Chapter LXXXXXXXI
474	Chapter LXXXXXXXII
475	Chapter LXXXXXXXIII
476	Chapter LXXXXXXXIV
477	Chapter LXXXXXXXV
478	Chapter LXXXXXXXVI
479	Chapter LXXXXXXXVII
480	Chapter LXXXXXXXVIII
481	Chapter LXXXXXXXIX
482	Chapter LXXXXXXXI
483	Chapter LXXXXXXXII
484	Chapter LXXXXXXXIII
485	Chapter LXXXXXXXIV
486	Chapter LXXXXXXXV
487	Chapter LXXXXXXXVI
488	Chapter LXXXXXXXVII
489	Chapter LXXXXXXXVIII
490	Chapter LXXXXXXXIX
491	Chapter LXXXXXXXI
492	Chapter LXXXXXXXII
493	Chapter LXXXXXXXIII
494	Chapter LXXXXXXXIV
495	Chapter LXXXXXXXV
496	Chapter LXXXXXXXVI
497	Chapter LXXXXXXXVII
498	Chapter LXXXXXXXVIII
499	Chapter LXXXXXXXIX
500	Chapter LXXXXXXXI

INTRODUCTION

Purpose

Beginning Applesoft BASIC is designed to provide classroom teachers with a fundamental background in BASIC programming, not to turn them into programmers or computer scientists. Why do teachers want to know how to program?

To share something which is intellectually stimulating and highly motivating with their students.

To avoid feeling "left behind" by their students, many of whom know how to program a computer.

To understand better how the programs teachers use in their classrooms are created and how those programs "make the computer do what it does."

To have fun.

MECC studies have found that if young students use computers to run only prepared computer courseware, they do develop an increased awareness for computer capabilities and social impact. Unfortunately, they also feel a higher level of "computer mystique" (finding the computer to be a mysterious, magical, "incomprehensible" box) than those with no computer experience at all. Students need to have the experience of commanding the machine to do things in order to take the mystery (if not all the magic) out of the machine.

It follows that teachers who have had a "software only" exposure to computers might also tend toward a computer mystique. Introductory courses in programming like this one are intended to make the computer more understandable, and therefore more friendly.

The primary audience for these training materials is the classroom teacher. The course has been designed, however, to met the needs and interests of a broad range of individuals, including teachers and non-teachers alike. In addition, some of the materials and activities included can be taken by training participants back to their schools for use with students.

Course Prerequisites

This course assumes no prior programming experience. It is assumed, however, that the class participants are familiar with the Apple computer and its operation with prepared software. If participants are not experienced with computers, additional preparatory sessions should be considered prior to using these materials. Some introductory activities designed specifically for teachers may be drawn from the MECC training manual Using Computers in the Classroom (Apple Version), #651.

Class Format and Setting

These materials are designed to be used in five sessions, each at least two hours in length. The suggested times for each activity, provided in the Instructor Notes, are minimums. A more comfortable schedule would allow two-and-a-half or more hours per session. The class could be offered during the academic year on an after-school basis once a week. Spacing class sessions several days apart allows class participants to experiment on their own and to assimilate each new major concept.

Because the hands-on component of this course is essential, it is suggested that sufficient computers be available in order to ensure there are no more than two participants per computer. It is an advantage if some computers in the room (or at least the demonstration computer in the front) are equipped with printers. Participants may wish to print-out program listings of large projects or exercises, especially before or after class.

The course structure alternates between computer activities of 20-40 minutes and informational sessions of 15-30 minutes. Because of this, it is advantageous to separate the hands-on lab area of the training room from the demonstration/lecture area. Ideally these two areas should be in the same room to allow rapid transition between the two class modes.

Why BASIC?

There are a variety of computer languages which may be used to meet this book's purpose. Logo, especially for teachers of elementary and junior-high-age students, may be preferred to fulfill the computer literacy goals intended for this course. BASIC is, however, a relatively simple language to use at an introductory level, an all-purpose language, and currently the most widely known and available computer language for personal computers.

Instructional Philosophy

Relevance - A primary educational motivator is relevance. The activities and examples developed in this course are intended to be simple yet useful applications of computer programming. The three most common applications of personal computer software are entertainment, personal and financial management, and education. Major projects in each of these three areas are developed by the participants during the course.

Graphics - Another motivational feature of this course design is the early introduction of graphics. The Apple's low-resolution graphics capabilities, along with the print output mode of the computer, are investigated during the first class session. Programming activities use both the computer's print and graphics capabilities to investigate each succeeding topic. Teachers will find the experience with graphics particularly valuable, since it is an area which is exciting to their students.

Hands-on - Programming, especially at the introductory level, is not a spectator sport. The participants can learn only by doing it. At the same time they need frequent informational breaks for general presentations, feedback, and questioning. Each class session (2 to 2½ hours) consists of several hands-on activity sessions separated by information breaks. During each class session the balance between hands-on activity and verbal information is roughly equal. End-of-session exercises (meant to be done as individual lab exercises or as homework) increase the ratio of hands-on to presentation activities.

How to build a program - This programming course is not intended to be a BASIC reference manual with narration and examples. For instance, not all the possible uses of the PRINT statement are developed in the first class session. Programming details are filled in as needed in the course or at the instructor's discretion.

In many courses where the participants have been directed to concentrate on the statements of BASIC, they have learned well how each statement works. However, they frequently have trouble putting more than a few statements together into a working program.

Rather than concentrating on an encyclopedic knowledge of syntax for each BASIC statement, the emphasis in this book is on the principal structures in programs (input, output, looping, and branching) and on constructing programs from the simple structures. To this end, several simple programs are developed which are continually extended as new programming structures are presented. This project approach gives the participants a good feeling for one way in which a program might be built, as well as giving concrete, relevant examples of using the most recently developed programming topic.

Objectives

In addition to the overall objectives mentioned in the preceding sections—including developing positive attitudes toward computing and the ability to construct a program—there are specific objectives related to the use of the BASIC statements and program structures for each class session. These objectives are listed in the Instructor Notes for each class session.

Class Materials

For each class session the instructor should have available the following for demonstrations and presentations:

- Apple II+ or Apple //e computer with disk drive
- Large color display monitor or projector
- Beginning Applesoft BASIC training diskette
- Transparency projector and/or blackboard

Materials for students include:

Activity handouts for each class activity
Beginning Applesoft BASIC Reference Materials
Beginning Applesoft BASIC participant diskette (prepared from the training diskette)

Especially during the first few sessions, it is advisable to hand out each activity separately. There is plenty for participants to explore without rushing into new material ahead of the trainer's guidance.

Preparation of Participant Diskette

The diskette accompanying this booklet may be copied for classroom use (and only for that purpose). The participant diskette may be prepared by copying the Beginning Applesoft BASIC training diskette, then deleting selected programs from the copy.

The training diskette contains sample programs for class demonstration and analysis by class participants. These have descriptive titles such as MOVING BOAT or TAX 1. These programs should remain on the participant diskettes.

Classroom activities are saved on the diskette by the activity number. For instance, the program that the students complete during Activity 10, part 2, is on the diskette as A 10-2. These "activity keys" should be deleted from the student diskette. (See catalogs of training diskette and participant diskette on page 5 for comparison.)

Example programs for the end-of-session exercises are also on the training diskette, labeled by the session number (Roman numeral) and, if appropriate, by exercise number (such as EX III-1). The instructor may opt to leave any or all of these on the student diskette in order to provide students with something to inspect in the event that they need help while doing an exercise away from the classroom.

CATALOG OF TRAINING DISKETTE:**PARTICIPANT DISKETTE:**

DISK VOLUME 254

I 000 [APPLESOFT BASIC FOR TEACHERS]

A 002 TAX 1

A 003 TAX 2

A 003 TAX 3

A 003 TAX 4

A 003 TAX 5

A 002 LONG GRAPH

A 008 BOOSHOO

A 005 VALENTINE

A 004 MOVING BOAT

A 004 FLASHY BOAT

A 005 BOAT AND ISLAND

A 002 A 2

A 002 A 3

A 003 A 4

A 002 A 5

A 002 A 6

A 003 A 7

A 002 A 8

A 002 A 9

A 002 A 10-2

A 002 A 10-3

A 002 A 11-2

A 002 A 11-3

A 003 A 11-4

A 003 A 12-1

A 003 A 12-2

A 008 EX I-1

A 002 EX I-2

A 003 EX II

A 002 EX III-1

A 002 EX III-2

A 006 EX IV

*B 010 LOGO

*A 008 HELLO

Demonstration
ProgramsActivity
Keys

Exercises

DISK VOLUME 254

I 000 [APPLESOFT BASIC FOR TEACHERS]

A 002 TAX 1

A 003 TAX 2

A 003 TAX 3

A 003 TAX 4

A 003 TAX 5

A 002 LONG GRAPH

A 008 BOOSHOO

A 005 VALENTINE

A 004 MOVING BOAT

A 004 FLASHY BOAT

A 005 BOAT AND ISLAND

*B 010 LOGO

*A 008 HELLO

ACKNOWLEDGEMENTS

These training materials were designed and written by Thomas Boe of the MECC staff. Paul Dillenberger, Minneapolis Public Schools, reviewed the materials and provided valuable suggestions.

Item	Description	Quantity	Unit Price	Total Price
1	Item 1 Description	100	1.00	100.00
2	Item 2 Description	200	2.00	400.00
3	Item 3 Description	300	3.00	900.00
4	Item 4 Description	400	4.00	1,600.00
5	Item 5 Description	500	5.00	2,500.00
6	Item 6 Description	600	6.00	3,600.00
7	Item 7 Description	700	7.00	4,900.00
8	Item 8 Description	800	8.00	6,400.00
9	Item 9 Description	900	9.00	8,100.00
10	Item 10 Description	1,000	10.00	10,000.00
11	Item 11 Description	1,100	11.00	12,100.00
12	Item 12 Description	1,200	12.00	14,400.00
13	Item 13 Description	1,300	13.00	16,900.00
14	Item 14 Description	1,400	14.00	19,600.00
15	Item 15 Description	1,500	15.00	22,500.00
16	Item 16 Description	1,600	16.00	25,600.00
17	Item 17 Description	1,700	17.00	28,900.00
18	Item 18 Description	1,800	18.00	32,400.00
19	Item 19 Description	1,900	19.00	36,100.00
20	Item 20 Description	2,000	20.00	40,000.00
21	Item 21 Description	2,100	21.00	44,100.00
22	Item 22 Description	2,200	22.00	48,400.00
23	Item 23 Description	2,300	23.00	52,900.00
24	Item 24 Description	2,400	24.00	57,600.00
25	Item 25 Description	2,500	25.00	62,500.00
26	Item 26 Description	2,600	26.00	67,600.00
27	Item 27 Description	2,700	27.00	72,900.00
28	Item 28 Description	2,800	28.00	78,400.00
29	Item 29 Description	2,900	29.00	84,100.00
30	Item 30 Description	3,000	30.00	90,000.00
31	Item 31 Description	3,100	31.00	96,100.00
32	Item 32 Description	3,200	32.00	102,400.00
33	Item 33 Description	3,300	33.00	108,900.00
34	Item 34 Description	3,400	34.00	115,600.00
35	Item 35 Description	3,500	35.00	122,500.00
36	Item 36 Description	3,600	36.00	129,600.00
37	Item 37 Description	3,700	37.00	136,900.00
38	Item 38 Description	3,800	38.00	144,400.00
39	Item 39 Description	3,900	39.00	152,100.00
40	Item 40 Description	4,000	40.00	160,000.00
41	Item 41 Description	4,100	41.00	168,100.00
42	Item 42 Description	4,200	42.00	176,400.00
43	Item 43 Description	4,300	43.00	184,900.00
44	Item 44 Description	4,400	44.00	193,600.00
45	Item 45 Description	4,500	45.00	202,500.00
46	Item 46 Description	4,600	46.00	211,600.00
47	Item 47 Description	4,700	47.00	220,900.00
48	Item 48 Description	4,800	48.00	230,400.00
49	Item 49 Description	4,900	49.00	240,100.00
50	Item 50 Description	5,000	50.00	250,000.00
51	Item 51 Description	5,100	51.00	260,100.00
52	Item 52 Description	5,200	52.00	270,400.00
53	Item 53 Description	5,300	53.00	280,900.00
54	Item 54 Description	5,400	54.00	291,600.00
55	Item 55 Description	5,500	55.00	302,500.00
56	Item 56 Description	5,600	56.00	313,600.00
57	Item 57 Description	5,700	57.00	324,900.00
58	Item 58 Description	5,800	58.00	336,400.00
59	Item 59 Description	5,900	59.00	348,100.00
60	Item 60 Description	6,000	60.00	360,000.00
61	Item 61 Description	6,100	61.00	372,100.00
62	Item 62 Description	6,200	62.00	384,400.00
63	Item 63 Description	6,300	63.00	396,900.00
64	Item 64 Description	6,400	64.00	409,600.00
65	Item 65 Description	6,500	65.00	422,500.00
66	Item 66 Description	6,600	66.00	435,600.00
67	Item 67 Description	6,700	67.00	448,900.00
68	Item 68 Description	6,800	68.00	462,400.00
69	Item 69 Description	6,900	69.00	476,100.00
70	Item 70 Description	7,000	70.00	490,000.00
71	Item 71 Description	7,100	71.00	504,100.00
72	Item 72 Description	7,200	72.00	518,400.00
73	Item 73 Description	7,300	73.00	532,900.00
74	Item 74 Description	7,400	74.00	547,600.00
75	Item 75 Description	7,500	75.00	562,500.00
76	Item 76 Description	7,600	76.00	577,600.00
77	Item 77 Description	7,700	77.00	592,900.00
78	Item 78 Description	7,800	78.00	608,400.00
79	Item 79 Description	7,900	79.00	624,100.00
80	Item 80 Description	8,000	80.00	640,000.00
81	Item 81 Description	8,100	81.00	656,100.00
82	Item 82 Description	8,200	82.00	672,400.00
83	Item 83 Description	8,300	83.00	688,900.00
84	Item 84 Description	8,400	84.00	705,600.00
85	Item 85 Description	8,500	85.00	722,500.00
86	Item 86 Description	8,600	86.00	739,600.00
87	Item 87 Description	8,700	87.00	756,900.00
88	Item 88 Description	8,800	88.00	774,400.00
89	Item 89 Description	8,900	89.00	792,100.00
90	Item 90 Description	9,000	90.00	810,000.00
91	Item 91 Description	9,100	91.00	828,100.00
92	Item 92 Description	9,200	92.00	846,400.00
93	Item 93 Description	9,300	93.00	864,900.00
94	Item 94 Description	9,400	94.00	883,600.00
95	Item 95 Description	9,500	95.00	902,500.00
96	Item 96 Description	9,600	96.00	921,600.00
97	Item 97 Description	9,700	97.00	940,900.00
98	Item 98 Description	9,800	98.00	960,400.00
99	Item 99 Description	9,900	99.00	980,100.00
100	Item 100 Description	10,000	100.00	1,000,000.00

This document is a statement of work for the project. It is not a contract and does not constitute an offer. It is intended to provide a general overview of the project and its objectives. The actual scope of work and the specific details of the project will be determined by the project manager and the client. This document is subject to change without notice.

Instructor Notes

BEGINNING APPLESOFT BASIC
Instructor Notes

Session I: Printing and Drawing - Output

Objectives:

To learn the functions of the BASIC commands dealing with output:

PRINT	HOME	GR	COLOR	PLOT
HLIN	VLIN	TEXT		

To learn the functions of the following Applesoft system commands:

NEW	LIST	RUN	SAVE	LOAD	CATALOG
-----	------	-----	------	------	---------

To learn how to edit a BASIC program.

Class Materials:

Activity Handouts 1 and 2 for each participant

Exercises I - "Output"

Participant versions of the Beginning Applesoft BASIC diskette
(See the "Preparation of Participant Diskette" section in the Introduction (p. 4)
for preparation details.)

Reference Materials

(The entire reference section should be distributed as a booklet to each training participant.)

Instructor Materials:

Beginning Applesoft BASIC training diskette

Transparency 1 - "Apple System Commands"

Session Outline:

Introductions (15 minutes)

Provide the necessary introductions if the participants are not familiar with the instructor, each other, the building, the classroom situation, etc. Acquaint the class with the goals of the course as outlined in the Course Overview section. In particular note that the intent is to acquaint them with the BASIC language, not to turn them into master programmers. Give them a realistic picture of what they may be able to accomplish by the end of the course.

Indicate that the three main uses of computers in the home are recreation, personal business and management, and education. They will not be able to create recreation programs such as Pac Man ®, but they will learn how to create graphics animations. They will not be able to create a Visi Calc ®, or Apple Business Graphics ® package, but they will create a checkbook balancing program and a program that could be used to graph financial information. They will not be able to create an elaborate educational application, but they will create a simple mathematics drill program, or may use the graphing program previously mentioned, or something similar, for educational purposes.

Run program A 12-1 from the Instructor copy of the Beginning Applesoft BASIC diskette to make this information more concrete. This is one of the programs that the participants will create as part of the class.

Introduce the participants to the project nature of the course. The goal of the course is to give the participants the background they need to create useful (relatively complex) programs. No complex program can easily be written in a single step. One method of developing complex applications that is very useful (especially for beginners) is to create a simple program, then add additional features and capabilities to it. This is a method that will be developed in this class.

The basic capabilities of the computer that the participants will learn how to employ include:

Output - putting out information, typing, drawing, etc.

Input - putting information into the computer

Looping - doing something repeatedly

Branching - doing different things under different conditions

The first class will be concerned with the first of these capabilities, output.

Activity 1: Giving the Computer Commands - pages 38-39 (50 minutes)

Preparation:

Hand out Activity 1, "Giving the Computer Commands." Point out that this first activity is a "discovery" approach to some of the Applesoft print and graphics commands. As such, no preparation for the activity is needed except for handling any mechanics concerning the use of the computers, and handing out the class diskettes.

Activity:

The participants should discover the functions of NEW, PRINT, HOME, GR, COLOR, PLOT, HLIN, and VLIN in this activity. Note that all of this activity is done in command mode, rather than as simple programs. The concept of a program is developed in the next activity. Those who finish early may explore the print and graphic capabilities further on their own.

Follow-up:

Have participants summarize what they learned about these BASIC statements. Points that should be emphasized include:

Math symbols: / and * may be new information for many who have had limited exposure to computers.

Graphics limits: Illustrate with a picture. Note that there is an error message when a horizontal value greater than 39 is used, but a vertical value between 40 and 47 just causes strange text characters below the picture. This range isn't "illegal," but rather is set aside for text use.

VLIN, HLIN: Have someone explain what the three numbers in each of these two statements mean, from their experience. If one thinks of a vertical line as a series of boxes, all are AT the same horizontal position on the screen. The other two numbers needed to identify the line segments are the vertical positions of its two endpoints.

Direct attention to pages R-10 and R-11 of the Reference Materials, which provide a summary of these output commands.

Activity 2: Writing a Program - pages 40-41 (25 minutes)

Preparation:

Discuss the idea of a program as a numbered set of commands that are performed in order.

Discuss some advantages of a program over "command mode," which the participants have just experienced. Some advantages include the abilities to do a series of commands many times without retyping them and to save a series of commands on a diskette for later use, as well as the fact that some of the later capabilities of BASIC (repeating, for instance) are only useful if a series of commands are in a program.

Demonstrate the command TEXT on the computer. Participants will need to type TEXT before starting the next activity so that they can see more than four lines of text at a time for program listings.

Activity:

There are some surprises in this activity. At the bottom of the first page participants are to predict what will happen after some graphics lines are added to their text program. Most will be surprised not to see the text at all. You may have to field this question during the activity time. Save involved explanations for the follow-up, after they have had time to think about it themselves for a while.

The function of NEW can be a bit of a surprise. Some class members will feel that they have done something wrong or that the NEW statement did "nothing" when the program no longer lists. Encourage them to type RUN to see what happens. What did NEW do to the program?

Follow-up:

1. Discuss the advantage of being able to RUN a program many times and being able to edit it.

Point out the three basic editing functions--deleting, replacing, and inserting lines.

"Play computer" with the prediction at the bottom of the first page of the activity.

Print THIS IS MY FIRST PROGRAM on the board; moving on to line 20, erase the board and put up a big rectangular "drawing area," plotting a point in it at about 2,23.

Why didn't they see their text? Because the computer covered it up with a blank graphics screen immediately after printing it. Indicate that one of the most important skills that one must develop in programming is putting oneself in the place of the computer, to "play computer" like this. In order to plan out a new program, or to discover the cause of an error in a program, this type of procedure is required.

2. You may wish to use Transparency 1, "Apple System Commands," to discuss the function of each of the commands experienced in this activity with a human analogy. Note that the program occupies a place in the "memory" of the computer. These commands are summarized for the participants on page R-1 of the Reference Materials Packet.

NEW may be looked at as giving the computer a "case of amnesia," wiping out everything it knew as program instructions so that the programmer can create a brand NEW program.

LIST is a listing out of what is in memory. RUN makes the computer follow the instructions stored in memory. These instructions (the program) usually includes input (like seeing and hearing in a human) and output (speaking or writing in a human).

The bottom half of the transparency shows the "diskette world." It is important to distinguish between what is in the computer's memory and what is on disk. Beginners will sometimes be afraid to type NEW for fear that a program that they have saved will be wiped off of the disk in addition to being cleared from memory.

Using the diagram, indicate that the diskette performs a similar function for computers as notebooks do for humans: some place to write information down in case it is "forgotten".

When a program is SAVED, the computer "writes" the program instructions onto the disk for later reference, as one might write easily forgotten information into a notebook.

When a program is LOADED the diskette is "read" and the program commands stored there are copied into memory. The computer can only "remember" one program at a time. LOADing a program (or typing RUN with a program name following) will erase whatever is in the computer's memory.

CATALOG should be demonstrated using the class diskette. Compare this to a table of contents of a well organized notebook.

After CATALOGing the diskette, choose a program from the catalog and demonstrate RUN followed by the program name. Explain this command as a short-cut which performs the LOAD and RUN functions simultaneously.

Exercises I Preparation - page 42 (5 minutes)

Review the Exercise assignment, making sure everyone feels confident about writing any series of commands preceded by numbers to make a program, and being able to save this program on a diskette. The participant work diskettes should be used to save these programs.

Exercises are supplied at the end of each class session. These may be done in an open laboratory or as a "homework" assignment. The purpose of such exercises is to provide extra practice, keep knowledge fresh between class sessions, and to provide a problem or two which can be investigated further without the time constraints imposed in a classroom setting.

BEGINNING APPLESOFT BASIC

Instructor Notes

Session II: Variables

Objectives:

To understand the use of variables as symbols for numbers (numeric variables) and sets of characters (string variables).

To know when to use LET or INPUT when assigning a value to a variable in a program.

Class Materials:

Activity Handouts 3, 4, and 5

Exercises II - "Variables"

Reference Materials

Participant diskette

Instructor Materials:

Beginning Applesoft BASIC training diskette

Transparency 2 - "The Bar Graph"

Session Outline:

Show and Tell (10 minutes)

Have a few of the participants show their projects from Exercise I Part 1. Some will have spent considerable time making elaborate graphics programs (see BOOSHOO, VALENTINE, and EX I-1 on the class diskette as examples of past projects).

Demonstrate how such programs may be shared with others without duplicating the work involved in coding them. Add a REM statement to a participant's program, noting the author's name. Discuss the purpose of REM statements. Put another diskette in the computer and SAVE the program. The program has now been "shared."

Respond to any general questions relative to the exercises.

Activity 3: Using Variables in a Sentence - page 43 (40 minutes)

Preparation:

1. Introduce the concept of a variable using programs TAX 1 and TAX 2.

Have participants examine the listing of TAX 1 from page R-3 of the Reference Materials and predict what the output will be.

Run the program to demonstrate.

Predict the output by placing the name NORBERT in the column marked NA\$ and the various numbers in the appropriate columns. Point out the difference between the labels of these boxes (variable names) and their contents. Also point out the difference between the characters which can be in AM (numbers only) and those in NA\$ (any group of characters).

Follow a similar procedure with TAX 2.

2. Discuss which of these two programs is more flexible and why. Distinguish between the programmer deciding what something should be (LET) and the program user deciding what it should be (INPUT). Since the tax rate (line 200) and the fact that total cost is equal to amount spent plus tax (line 250) rarely change, that information can be built into the program by the programmer. It would be cumbersome to edit the program, however, each time a different amount was purchased (line 150), or a different customer came up to the check-out counter to make a purchase (line 100). Here it makes more sense to let the program user type in the information.

Introduce the idea of a prompt line. Point out that each of the INPUT statements in program TAX 2 are preceded by a PRINT statement. The INPUT causes a question mark to be printed on the screen, but without a message to precede it the program user is at a loss as to what he or she might be expected to do next.

Hand out Activity 3 and indicate that the first programming problem includes INPUTs preceded by prompt lines—that is, PRINT statements.

Activity:

The only thing that may need comment during this short laboratory activity is the use of semicolons and the need for spaces as indicated in some of the PRINT statements.

Follow-up:

Respond to any specific questions. Be sure that the participants understand:

- why NA\$ was used in this case (instead of NA);
- what the semicolon does in a PRINT statement;
- the purpose of the spaces in the PRINT statements.

Activity 4: Make Up a Story - page 44 (30 minutes)

Preparation:

Explain this activity briefly. Point out that the four questions and answers should require eight lines, four prompts (PRINT), and four responses (INPUT). Each of the four INPUTs should use a different variable name (like AN\$ for the animal). The final sentence can be done with a single PRINT statement similar to that in line 150 from the previous activity.

Activity:

Watch carefully during the laboratory activity for people who may be trying to use "THE FEROCIOUS" as a prompt for animal, "WALKED PAST" as a prompt for the "things", etc. Clarify the difference between the input phase of this program and the final product (a "surprise" sentence). Caution participants about spacing within the quotes before and after variables in the last line. It is easy to miss a space, but the resulting run-together words are very obvious.

Follow-up:

Respond to any questions about the second activity and have one of the class demonstrate the final product. Program A 4 on the training diskette is one solution to this activity.

Activity 5: The Bar Graph (Phase 1) - pages 45-46 (35 minutes)

Preparation:

Use Transparency 2, "The Bar Graph," to analyze the problem presented in the activity.

This is the first program that the class has seen which will be extended into a major project. Since they will be adding to the program several times, they should start this program at line 1000. That should give them plenty of room to add more lines wherever they desire. Each of the steps in the last paragraph of the activity should be the focus of their work as they begin. Each numbered item corresponds to a single program line. With the line-plotting command already provided, the entire program may be written in five lines. An additional line to HOME the screen might make the program look more polished.

Activity:

Direct the participants to look at the numbered steps in the last paragraph of the activity. They should then have little difficulty creating this first graphing program. Some will notice that most bar graphs have more than one bar and may begin to extend their program to two and three bars by simply repeating the prompt, input, and plotting statements. This is natural if they have time. However, tell them not to do too much work; they'll see an easier way to repeat the process during the next class session! Be sure that everyone saves a copy of this program on their diskettes, since they will need it for later versions of the graphing program.

Follow-up:

Make sure that everyone has completed a working version of the graphing program. Have a class member load and run the program and list it for the class. Program A 5 on the training diskette is one solution to this activity.

Exercises II Preparation - page 47 (5 minutes)

The exercise activity is a real-life use of simple programming with variables. The activity is not unlike the animal story, except that form letters are more generally useful (if not as entertaining). If possible bring an example of this kind of form letter to class, the kind that has your city and street embedded in sentences in the middle of the text.

Discuss why ZAP TOOTHPASTE should be set with a LET statement, whereas the name and title of the recipient should be accepted as input. Each of the lines of the letter should be a separate PRINT statement. None of the lines are over 40 characters long.

BEGINNING APPLESOFT BASIC
Instructor Notes

Session III: Repeating

Objectives:

To know the correct use of GOTO and FOR NEXT to repeat program operations.

To be able to use the counter in a FOR NEXT loop to perform such functions as graphing, doing animation, and creating tables.

Class Materials:

Activity Handouts 6, 7, and 8

Exercises III - "Repeating"

Reference Materials

Participant diskette

Instructor Materials:

Beginning Applesoft BASIC training diskette

Transparency 3 - "Looping"

Session Outline:

Show and Tell (5 minutes)

Have a class member demonstrate the form-letter project. See if anyone had any extensions to the project or questions.

Activity 6: Doing the Same Thing Over and Over - pages 48-49 (50 minutes)

Preparation:

One of the capabilities of the computer that everyone should know about is its ability to do the same task over and over again. The focus of this class session is to see two ways of doing this and to see some uses for this ability. Use Transparency 3, "Looping," when discussing these two loop forms.

1. GOTO loops:

Demonstrate Activity 6, Step 1 as an example of a simple GOTO loop.

Demonstrate TAX 3 as an additional example of a GOTO loop. A listing of TAX 3 may be found on page R-4 of the Reference Materials. Show that it does the same thing as the name loop in Activity 6; it repeats forever. The only difference between them is that TAX 3 repeats more than one line.

2. FOR NEXT loops:

Demonstrate Part 2 of Activity 6. Put the FOR on line 90 and the NEXT on line 200. Later in the activity the class participants will have to add lines before 90 to ask for address, city, etc., and then will add lines within the loop in order to print out all this information ten times in mailing-label format.

"Play computer" with the FOR NEXT loop, indicating the computer's process of adding one to the counter each time it goes through the loop and checking to see if it has exceeded the maximum. FOR NEXT is the best way to do a repeated function whenever something is to be repeated a given number of times.

Demonstrate TAX 4 and compare it to TAX 3. This is a more complicated situation than that in Activity 6, Part 2. Here the maximum for the counter is a variable, CU. The program user can decide how many times the tax calculation should be repeated.

Before beginning the activity, make sure that participants understand that Part 4 of the activity is to print out 10 labels from the same information (mass production), not to ask for the information 10 times, printing out a different label each time! Also be sure that they start the activity at Part 1 even though the beginning of the activity has been used as a demonstration.

Activity:

The most common problem faced in this activity is figuring out which lines need to be repeated, then figuring out where to put them. Guide the class by telling them that they need to be concerned about what exactly they would like to repeat before they get started. This becomes critical in Parts 3 and 4 of the activity. In this activity, input should not be in the loop; rather, output should. The questions should only be asked once; the mailing labels should be printed 10 times. Show them how to bracket the parts of the program which they want repeated, with the FOR at the top and the NEXT at the bottom.

Follow-up:

Demonstrate one labeling program from the class and respond to any questions.

Activity 7: Using the Counter Number in Graphics - page 50 (25 minutes)

Preparation:

Point out that the counter number can be used for doing functions inside the loop other than just counting.

1. Type in the following program:

```
100 FOR C = 1 TO 15
110 PRINT C
120 NEXT C
```

In this example the counter itself can be printed out. Change line 110:

```
110 PRINT C , C * .04
```

A sales tax table has been created, based on the tax rate from the various TAX example programs (4%). Note the use of the comma. This is the first time that the class has seen this. Indicate its use for creating columns of information.

2. Edit the tax program above to look like this:

```
90 GR
100 FOR C=1 TO 15
105 COLOR=C
110 VLIN 0,39 AT C * 3
120 NEXT C
```

Save this program with the name COLORS on the training diskette for reference during Activity 8.

COLORS uses the counter number for two functions. First, the COLOR is set to the counter number, allowing a display of all the Low-Resolution graphics colors. Secondly, the counter C is multiplied by 3 to give the horizontal position of the color bar.

In the next activity the counter number is used for positioning colored lines on the screen in a similar manner to create a sailboat scene. This sailboat scene will be extended in later activities, including one of the exercises for this session.

Activity:

Little additional guidance is needed for this activity. Be sure that participants see the way that the loops are used to "fill in" the sky and the sea.

Follow-up:

"Play computer" with this program, and on the blackboard or overhead projector fill in a few lines of sky.

Activity 8: The Bar Graph (Phase 2) - page 51 (20 minutes)

Preparation:

Show LONG GRAPH, a program which extends the bar graph program laboriously by repeating all the steps. This program was done by a person who didn't know about loops. Activity 8 uses a FOR NEXT loop to simplify this process.

Examine Activity 8 and note that the counter number (MO) is used to place the bar. Just as in the COLORS example, the bars are placed three units apart by multiplying the counter by 3. LOAD, LIST, and RUN the COLORS example again for comparison.

Activity:

By referring to the COLORS example, the class should have little difficulty extending the bar graph program to take repeated inputs and create a bar for each. Again, the most common difficulty is in figuring out exactly what needs to be repeated. Some will perhaps include the GR command within the loop, causing the screen to clear for each input. Be certain that the class members SAVE this program, since they will need it for later embellishments.

Follow-up:

Demonstrate a copy of the graphing program at this point and respond to any questions.

Exercises III Preparation - page 52 (10 minutes)

The exercises are concerned with a repeated function that is used extensively in recreational and educational programs: animation.

Load EX III-1 from the training diskette and demonstrate it. This is the same as Part 1 of the exercises.

Go through the listing and discuss how the animation is carried out by successively drawing and then erasing (with the background color) the object which is to be "moved." This version of the program goes pretty fast.

Run EX III-2, which is a slower, smoother animation. Tell participants that this is the same as the program they will create in Part 2 of the exercises. It will be discussed at the beginning of the next class.

BEGINNING APPLESOFT BASIC

Instructor Notes

Session IV: Making Decisions

Objectives:

To know how to use IF THEN statements to make controlled branches in programs.

To use tests involving $>$, $<$, $>=$, $<=$, and $<>$ in IF THEN statements.

Class Materials:

Activity Handouts 9 and 10
Exercises IV - "Making Decisions"
Reference Materials
Participant diskette

Instructor Materials:

Beginning Applesoft BASIC training diskette
Transparency 4 - "Branches"
Transparency 5 - "Scaling The Graph"

Session Outline:

Show and Tell (10 minutes)

1. Run a participant's dot animation program or EX III-2 from the training diskette. Examine the listing with the class. Ask what the short loop inside the main loop is doing. (It's a "do-nothing loop," a timing loop inserted to waste some time and slow the animation.)

Delete the loop from this spot and add it right after the erasing portion of the animation. "Will the program work the same way?"

Demonstrate it. This time the waiting occurs when the box is erased and the animation blinks noticeably.

Discuss briefly the concept of nested loops. One loop can be completely inside another, as in this example, but it cannot "lap" across another loop.

2. Run MOVING BOAT and examine the listing from the example programs packet. This animation is similar to EX III-2 except the object is moving horizontally and is much more complex. Because drawing and erasing the whole boat would have "flashed" the picture too much, the programmer chose to draw the new boat directly over the old one, one square farther to the right, erasing just the trailing edge of the boat in blue as it moved along.

Demonstrate FLASHY BOAT, which does erase and redraw the whole boat each time it is moved.

Activity 9: The Bar Graph (Phase 3) - pages 53-54 (60 minutes)

Preparation:

Compare the listings and the operation of TAX 4 and TAX 5 (Reference Materials, pages R-4 and R-8). TAX 4 contains a FOR NEXT loop, which is fine if you know exactly how many times you wish to perform an operation. TAX 5, on the other hand, can check each time after a customer has been helped to see if there is anyone else waiting to be helped. This is an example of a backward branch using IF THEN. IF the "test" portion of a branch is true, THEN it branches to the indicated line. If not, it goes to the next numbered line in the program. This is a backward branch, which can be used to do looping. It is also possible to do forward branching to "jump over" part of the program.

Use Transparency 4 as an aid in discussing branching. In the following activity, class members will add forward and backward branches to their graphing programs.

Mention mathematical tests other than = that can also be performed. In the activity they will use Greater Than (>) and Less Than (<) as tests. List these on the board or overhead for the class, as well as >=, <=, and <>. Also mention that the action portion of the IF THEN statement is usually a line number, but it could be any command such as

```
IF YR = 21 THEN PRINT "THAT'S OLD ENOUGH!"  
or  
IF AN$ = "STOP" THEN END
```

Point out that there are three parts to the next activity. The first allows the graph to be used for input greater than 40, up to some maximum determined by the program user. This portion uses a scaling formula which may be discussed after the activity; it is not an important programming concept, only a useful mathematical extension to the program. Parts 2 and 3 of the activity require that the participants add forward and backward branches to the resulting scaled graphing program to handle "unusual" things that might be typed in by the program user. "Input processing" of this kind is a typical use of IF THEN branches in programs.

Activity:

There are not many lines to add in this section, but deciding on the placement and form of the IF THEN statements will take some time.

Point out that the problem with small inputs to this program, dealt with in Part 2 of Activity 9, is not confined to inputs of 0 or less. If the maximum is set for 1000, an input of 5 should round to produce zero boxes on the bar graph. Since the bar is produced with an HLIN command, the computer errors out trying to do this. The test for a value too low to plot is whether RT, the right end of the bar, is less than 0, the lowest possible low-resolution box.

Follow-up:

1. Have someone demonstrate and list a completed program which scales and handles high and low input. Respond to any questions about the branching.
2. Some participants may be curious as to how the scaling formula for RT presented in Part 1 of this activity works. Based on the interest and mathematical competence of the group, the instructor may wish to use Transparency 5 to discuss its operation. Emphasize that this line ($RT = AM/MX * 40 - .5$) was added to make the program much more useful, but it is not a critical programming concept. You may wish to describe the following steps to explain this process:

(AM/MX) is the fraction of the screen width that the bar covers. If AM is 40 and MX is 80, then the bar will be half the width of the screen.

The number 40 is the number of squares across the screen. By multiplying the fraction calculated above ($\frac{1}{2}$) by 40, the number of squares that need to be filled in by the bar is produced (20).

The instructor may wish to state that subtracting .5 in the final step is used to "round" to a whole number.

Subtracting .5 actually performs two functions here. In phase 1 of the bar graph, 1 had to be subtracted from the input value. Since the result now is no longer necessarily a whole number of boxes, we need to "round up." This can be accomplished by adding .5. Subtracting 1 and adding .5 is the same as subtracting .5.

Transparency 5 may also be used to discuss the problems of input that is too low or too high.

Activity 10: Multiplication Drill (Phase 1) - pages 55-56 (45 minutes)

Preparation:

Tell the class that they will create a common type of instructional program, a drill and practice program. This will be embellished further during the next class session, but for now it will pose multiplication problems in order, using only the multiples of 9. If they have time they can extend this to multiples of any number in optional Part 3 during this class session or on their own. It is important to concentrate on the function of the branches in the program when the user gets the problems right or wrong.

Activity:

Parts 1 and 2 of this activity may be completed in about 20 minutes. The only part in which the participants need to do some creative thinking involves placing a GOTO in order to avoid the "RIGHT" response after the program gives negative feedback for a wrong answer.

Some may have time to try Part 3. Basically, all that is needed here is a prompt line, an INPUT line, and the change of the number 9 in lines 120 and 160 to a variable. Seeing what is needed, however, is the hard part. Be available to do some guiding for those who need help on this.

Follow-up:

Load in a participant's program in which Part 3 has been completed or use A 10-3 from the Instructor's diskette.

Go over the function of the branching. Pay special attention to the need for the GOTO.

The diagram of the forward branch, Transparency 4, can be used to show that this is a problem common to most forward branches. Unlike the branches of a tree, a branch in a BASIC program always comes back to the "main stem." The GOTO is necessary to totally isolate the FALSE case from the TRUE case. One can think of this as a "collision" between the two possible program paths. The GOTO is the only way to avoid such collisions.

Indicate what changes were needed in order to extend the basic program to be able to test the multiples of any number. Test out the program by asking for multiples of 100.

Mention that it is relatively easy to have the program print out random problems. If there is extra time you may wish to discuss a randomizing formula with the class or with interested individuals:

```
A = INT (RND (1) *10 + 1)
```

This will make A a random number from 1 to 10. If B is created the same way, the program they have can be modified to print out the problem like this:

```
PRINT A;" x "; B;" = ";
```

and check the answer like this:

```
IF AN = A * B THEN 230
```

Exercises IV Preparation - pages 57-58 (15 minutes)

Prepare the class for the exercise activity by running the program BOAT AND ISLAND from the class diskette. Their task is to make this animation more realistic by sinking the boat IF it hits the island, keep it moving IF it is to the left or right of the island, or keep the boat beached IF it starts on the island. They also will change the first few lines so that the island and boat positions may be INPUT into the program.

BEGINNING APPLESOFT BASIC

Instructor Notes

Session V: Accumulators, Counters, and Flags

Objectives:

To learn the concepts and use of accumulators, counters, and flags in programming.

Class Materials:

Activity Handouts 11 and 12
Reference Materials
Participant diskette

Instructor Materials:

Beginning Applesoft BASIC training diskette

Session Outline:

Show and Tell (10 minutes)

Demonstrate and list someone's SINKING BOAT program. Try out all cases—boat to the left, boat to the right, and boat on top of the island.

Review each of the required program additions.

Activity 11: The Electronic Checkbook Record - pages 59-60 (60 minutes)

Preparation:

No new BASIC statements will be learned during this class session. New uses for the old statements will be covered. The main project of the session is to create a complete checkbook balancing program. In doing this, three new concepts will be developed: accumulator, counter, and flag.

Point out lines 1080 and 1040 on the first page of Activity 11. `LET BA = BA + AM` may seem like a rather strange mathematical statement at first. Explain how such statements work.

BA is an accumulator, something to which things are added or subtracted. This is appropriate for a checkbook balance (one might hope to accumulate, anyway!).

CK in line 1040 is a counter, a special kind of accumulator which adds a given amount each time a loop is executed. The counter in the FOR NEXT loop is a special example of a counter.

Activity:

Activity 11 is rather self-explanatory. While testing out Part 2 some participants will object that checks are used up even when they are making a deposit. Direct their attention to Part 3, which takes care of that flaw.

Follow-up:

Run and list a participant's program as a demonstration. Trace the program steps for a negative input, a positive input, and an input of 999999.

Have participants suggest further improvements in the program (perhaps the program could ask if the entry is a check (C), a deposit (D), or a voided check (V)).

Activity 12: More Accumulating and Counting - page 61 (50 minutes)

Preparation:

Accumulators can be used to create totals, averages, and percentages. The following activity will use an accumulated total to calculate an average and will graph that average. It is the final elaboration on the graphing program.

Activity (Part 1):

Once they have created statements to calculate the total, participants may be at a loss as to where the average calculation should go. Some will try to adapt the program line that graphs the rest of the data bars so that it will graph the average at the end. Although this is possible, it is easier just to calculate the average and plot the final bar after the loop is finished with all 12 cycles. In order to be plotted, the value for the average needs to be scaled (a value of RT calculated for it). Finally, the same HLIN command used inside the loop to do the plotting can be used after the loop for the average. Make sure that participants save this final version of the graphing program.

Follow-up (Part 1):

Respond to any questions about using the accumulator to calculate and plot an average.

Introduce the final activity as a use of an accumulator in the participants' previously-created instructional program. This time they have the ability to add up (accumulate) the number of problems done correctly. Finally, from the total correct they will print out a percentage score.

Activity (Part 2):

The major challenge of this activity is to place the accumulator line in a spot in the program where any correct answer will add to it, but any incorrect answer will not. Some participants may have to experiment a bit with this. Again, this is a fine place to encourage people to "play computer."

Conclusion (5 minutes):

Review the class goals and what has been accomplished by the class members:

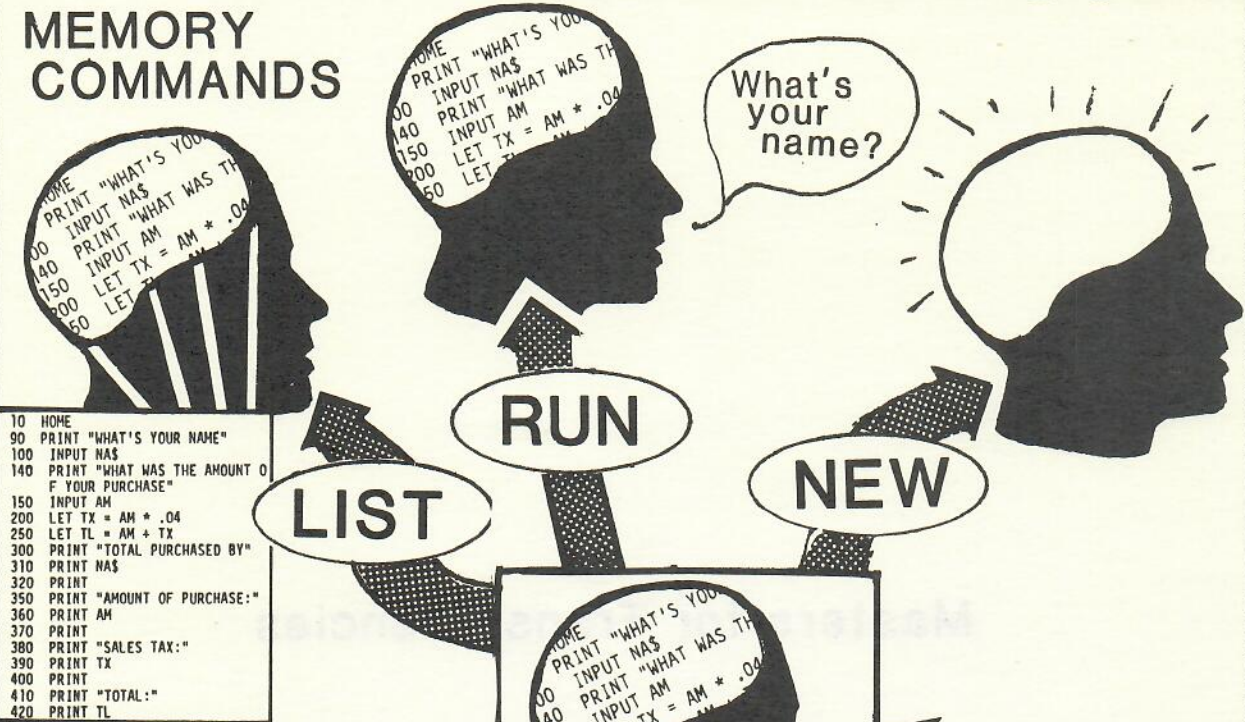
1. Producing print and graphics output;
2. Storing information (numbers and words) in the computer;
3. Investigating the ability of computers to perform repetitive tasks;
4. Investigating the ability of computers to "test and decide";
5. Learning how totals and averages are calculated and stored;
6. Building complex programs from simpler ones;
7. Seeing first-hand how some familiar computer applications work, including animation, graphing, form letters, accounting, and drill programs.

Point out any future directions that class members might take to get more advanced instruction in BASIC or in related topics.

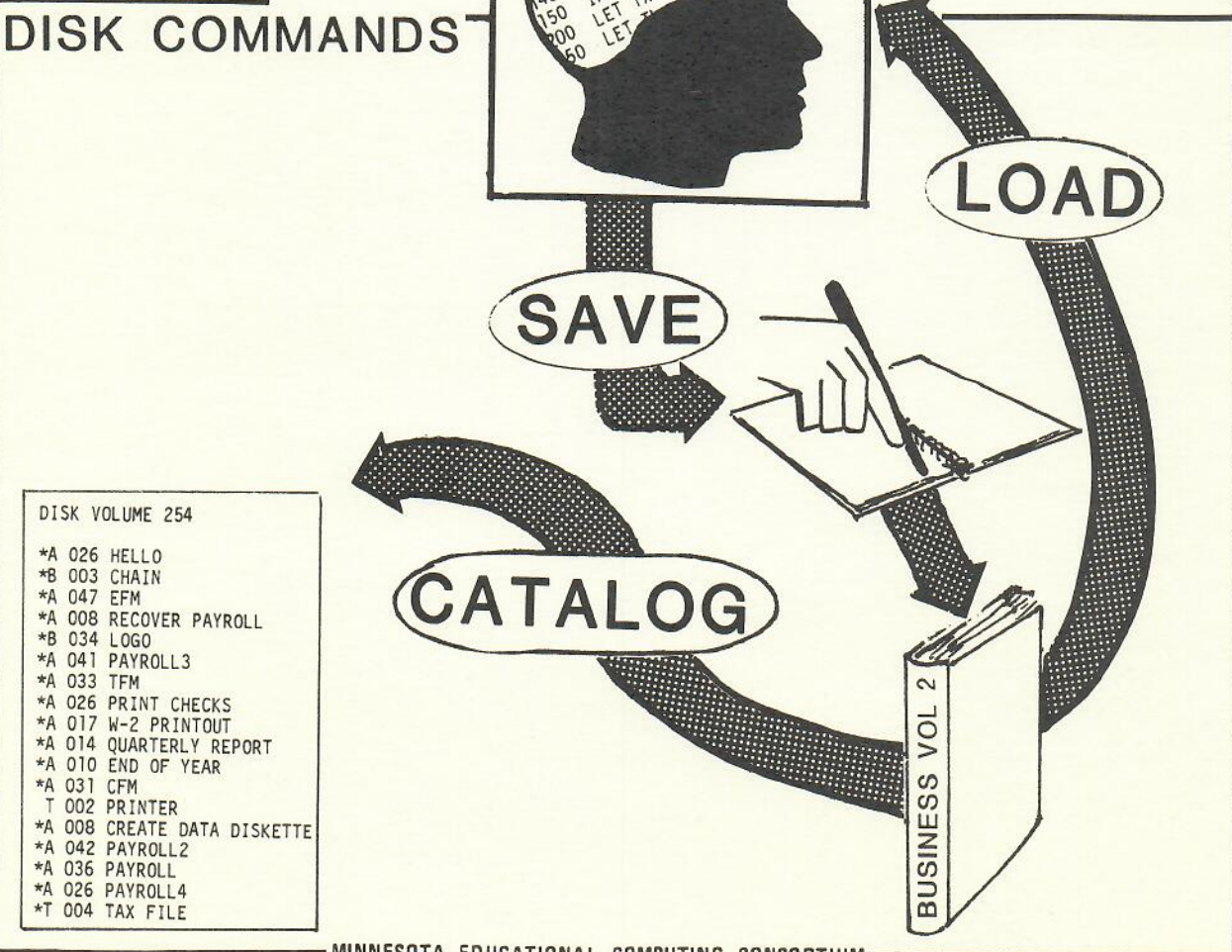
Masters for Transparencies

APPLE SYSTEM COMMANDS

MEMORY COMMANDS



DISK COMMANDS



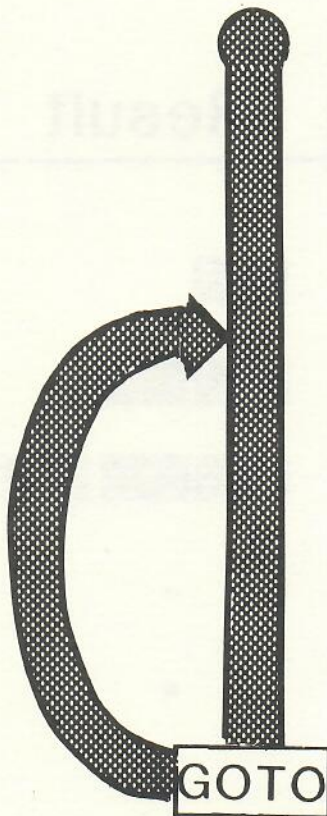
THE BAR GRAPH

AM	END POINTS		Result
	Left	Right	
1	0	0	
2	0	1	
3	0	2	
.	.	.	.
.	.	.	.
.	.	.	.
39	0	38	ETC.
40	0	39	

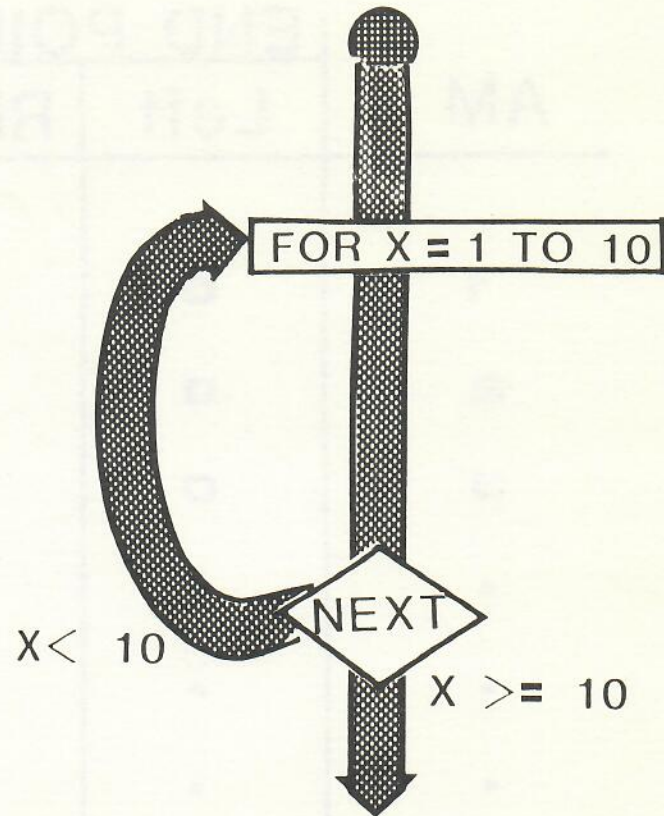
HLIN 0, AM-1 AT 19

LOOPING

INFINITE

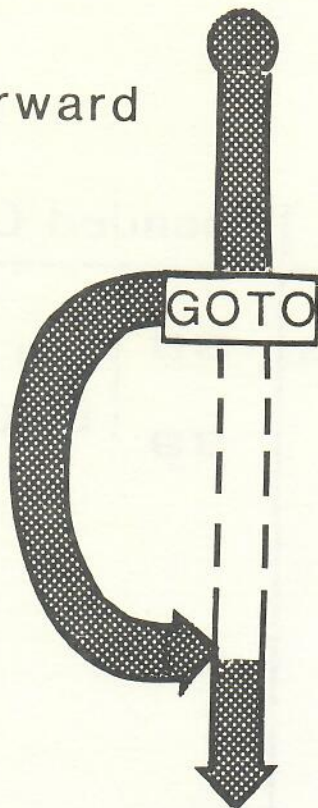


FINITE

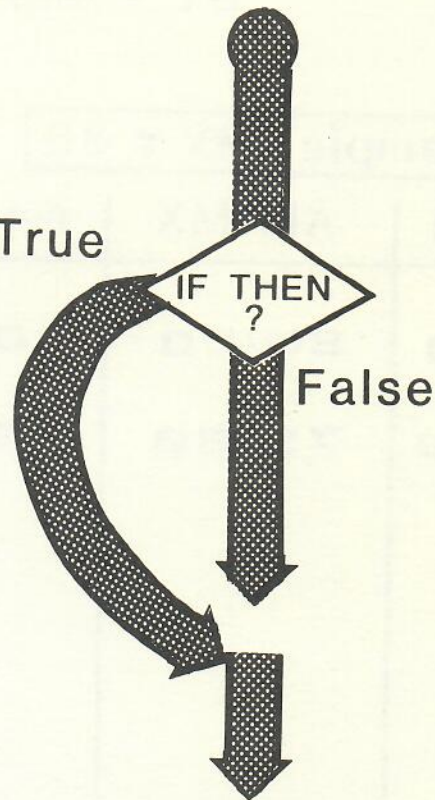


BRANCHES

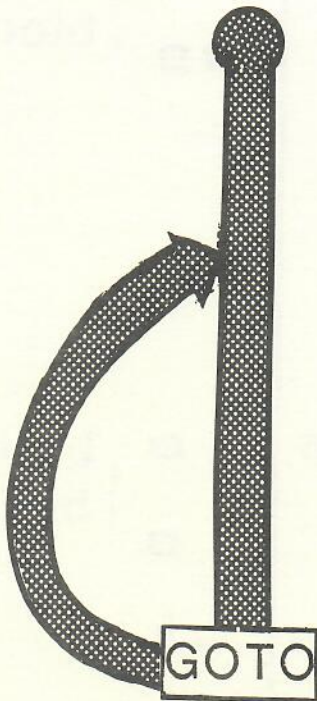
Forward



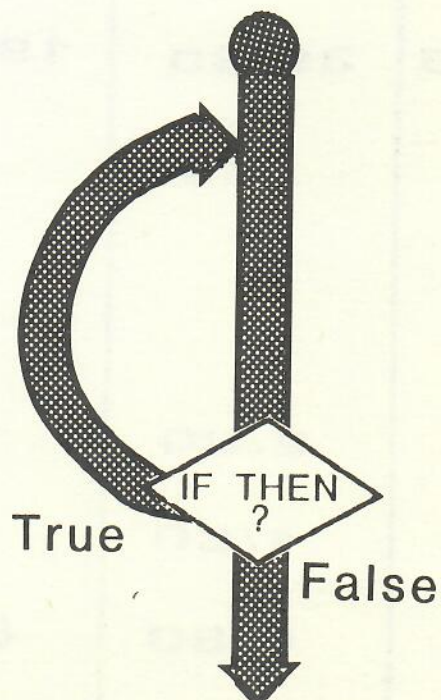
True



Backward



True



SCALING THE BAR GRAPH

$$RT = AM/MX * 40 - .5$$

Example: $MX = 80$

AM	AM/MX	*40	-.5	Rounded Down
80	80/80	40	39.5	39
79	79/80	39.5	39	39
.				
.				
.				
40	40/80	20	19.5	19
39	39/80	19.5	19	19
.				
.				
.				
2	2/80	1	.5	0
1	1/80	.5	0	0
0	0/80	0	-.5	-1

} top
block

} middle
block

} bottom
block

Masters for Activity Handouts

Printing and Drawing - Output
Giving the Computer Commands

1. Printing Out Things

Put in your diskette and start up the computer system.

Type NEW
Type HOME

Type the following lines, including the word PRINT and the quotation marks (press the RETURN key at the end):

PRINT "MY NAME IS OSWALD"
PRINT "NO IT IS NOT"

What does the computer do with a PRINT command followed by a statement in quotes? _____

Maybe the quotes aren't needed. Type this:

PRINT MY NAME IS OSWALD

What did the computer type back to you? _____

Here are some other PRINT statements without quotes. Record what is printed by the computer in each case:

TYPE:	COMPUTER PRINTS OUT:
PRINT 5 + 7	_____
PRINT 1000 / 45	_____
PRINT 1000 * 45	_____

What does the computer do with a PRINT command followed by a statement not in quotes? _____

Things are pretty messy on your TV screen by now.

TYPE: HOME

What does HOME do? _____

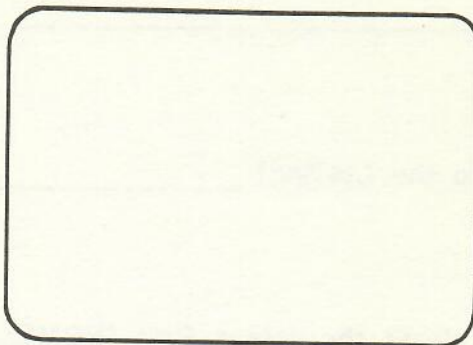
Printing and Drawing - Output
Giving the Computer Commands2. Drawing Out Things

Type the following 3 lines:

```
GR
COLOR=2
PLOT 20,20
```

GR changes the top of the screen to a "drawing pad" (G^Raphics). COLOR picks one of 16 possible colors (0-15). This choice may not be too exciting on a black and white TV! PLOT plots a point.

On the picture of the TV screen below draw and label the point (20,20) that we plotted.



PLOT some more points like (20,20), (0,0), and so on, drawing and labeling their positions on the picture above.

Label the picture to show where printing appears.

What is the largest horizontal number value you can use? _____

What is the largest vertical number value you can use? _____

Maybe you're tired of doing points. Here's how to do vertical lines (VLIN) and horizontal lines (HLIN):

Type: VLIN 0,20 AT 39

Type: HLIN 10,30 AT 0

Draw and label these two lines on the screen picture above. From these two examples decide what the three numbers stand for in the two commands. Write your conclusions in the space below:

Printing and Drawing - Output
Writing a Program

Type: NEW

Type the following 2 lines (including the numbers):

```
10 PRINT "THIS IS MY FIRST PROGRAM"  
20 END
```

Type: LIST

What does LIST do? _____

Type: RUN

Type: RUN

What does RUN do? _____

Type: 5 HOME

Type: LIST

Where did line 5 fit into the LISTing? _____

Type: RUN

This time the program clears the screen first (HOME), then prints out the message.

```
Type: 20 GR  
30 COLOR=2  
40 PLOT 2,23
```

Type: LIST

What happened to the line: 20 END? _____

Predict what will happen when you type RUN: _____

Type: RUN

What did happen? _____

BEGINNING APPLESOFT BASIC

Activity 2

Page 2

Printing and Drawing - Output Writing a Program

Type: TEXT

Type: 10

Type: LIST

What happened to line 10? _____

Type: 50 PRINT "THIS IS MY FIRST PROGRAM"

Type: LIST

Predict what will happen when you type RUN? _____

TYPE: RUN

What did happen? _____

Type: SAVE PROGRAM 1

Type: TEXT

Type: LIST

Is your program still in the computer's memory? _____

Type: NEW

Type: LIST

Is your program still in the computer's memory? _____

What does the NEW statement do? _____

If you have extra time, begin the exercises for this section.

BEGINNING APPLESOFT BASIC

Printing and Drawing - Output Exercises I

1. Graphics Output

Using the graphics commands GR, COLOR, PLOT, VLIN, and HLIN write a program which will draw something of your choice on the screen using at least two colors. It could be a face, a house, a word (in large colored letters), or anything else that strikes your fancy. Save this program on your diskette.

You may wish to duplicate the "Low-Resolution Graphics Planning Sheet," page R-14 of the Reference Materials, to use when designing your picture.

2. Printed Output

Write a program which has at least one PRINT statement in which the computer is acting like a typewriter (uses quotes) and one PRINT statement in which the computer acts like a calculator (without quotes). Example outputs might include:

123 TIMES 456 EQUALS:	no calculation done in the PRINT here
-----------------------	--

56088	The computer calculates the answer in this PRINT statement
-------	---

THE NUMBER OF SECONDS IN A DAY IS:

86400

You only need PRINT statements to write this program. You may want to use HOME also. Save this program on your diskette.

Variables
Using Variables in a Sentence

Type: NEW

Type in the following program:

```
10 HOME
100 PRINT "WHAT IS YOUR NAME"
110 INPUT NA$
120 PRINT "NICE TO MEET YOU,"
130 PRINT NA$
```

RUN the program.

Now change lines 100 and 120 like this:

```
100 PRINT "WHAT IS YOUR NAME ";
120 PRINT "NICE TO MEET YOU, ";
```

notice the extra spaces
and the semicolons



Add the following lines to the program:

```
115 PRINT
140 PRINT
150 PRINT "I THINK "; NA$; " IS A GREAT NAME."
```

spaces again!



What does the semicolon (;) do in a BASIC PRINT statement?

Variables
Make Up a Story

Type: NEW

Write a program that asks for an animal, then a number, then some "things", then a color:

Computer types these.

Person types (inputs) these.

TYPE IN AN ANIMAL:

RABBIT

TYPE IN A NUMBER:

48

TYPE THE NAME OF SOME THINGS (PLURAL):

LAMP POSTS

TYPE IN THE NAME OF A COLOR:

CRIMSON

The program should then write out a sentence like the one below, filling in the blanks using the words that were typed in by the person as in the example above:

THE FEROCIOUS (animal) WALKED PAST (some number) SMALL (things)
THAT TURNED (color) WITH FRIGHT.

SAVE this program on your diskette.

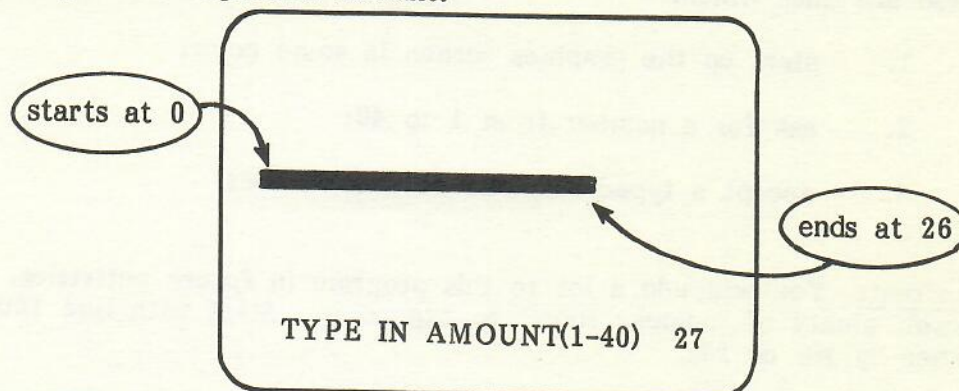
Variables
The Bar Graph (Phase 1)

Type: NEW

One piece at a time you will build a program that will create a bar graph for income (or expenses, or anything else) for a 12-month period. This program is your first step in that project.

You will write a program which will ask you to type any number from 1 to 40. The computer will then create a horizontal bar of that length in the middle of the screen.

In a bar graph you want the bars to start at the left of the screen and go to the right the requested amount:



By creating a table, we can figure out a formula for the HLIN command necessary to make this bar graph work:

input value # of blocks desired	HLIN end points	
	left	right
1	0	0
2	0	1
.	.	.
.	.	.
.	.	.
39	0	38
40	0	39

It starts at 0 and goes to one less than the number typed in!

Variables

The Bar Graph (Phase 1)

One of the lines in your program will be

HLIN 0, AM - 1 AT 19

where AM is the amount that has been typed into the computer. This formula gives the proper values for the right endpoint of the line for any number from 1 to 40 which is typed in. The horizontal line is plotted in the center of the screen (the vertical position is 19).

You now know one of the program lines you need to type in. All you have to add are lines which:

1. start up the graphics screen in some color;
2. ask for a number from 1 to 40;
3. accept a typed response as that number.

Important: You will add a lot to this program in future activities. Give yourself plenty of "number room" to expand it. Start with line 1000 and number by 10s or 20s.

Save this program, giving it a name like GRAPH 1.

BEGINNING APPLESOFT BASIC

Variables Exercises II

INPUT: The Form Letter

Some companies send out computerized form letters that look as if they were written especially to you. The letter below is an example:

DEAR MS. GREEN,
YOU MAY HAVE HEARD ABOUT OUR PRODUCT
ZAP TOOTHPASTE. WE'RE SO SURE THE WHOLE
GREEN FAMILY WILL LOVE ZAP TOOTHPASTE
THAT WE'RE SENDING YOU A FREE SAMPLE
WITH THIS LETTER.

YOURS TRULY,
MARY SEARS
ZAP TOOTHPASTE REPRESENTATIVE

Create the letter above on your computer. The underlined parts should be easily changed for letters to different people or for different products. The family name will change for every letter you send out, so make that an INPUT question before the letter is printed. You will also want to be able to change the title (Mr., Ms., Mrs., or Miss.)

The product name, ZAP TOOTHPASTE, won't change frequently, so don't INPUT that each time you want to print out a letter. You should, however, store ZAP TOOTHPASTE in a variable using LET. Who knows when you might want to use this same letter for another of your company's products, like BIF DOGFOOD?

Try out the program with at least two different family names. Finally, change the product name to BIF DOGFOOD to see if that works as well.

SAVE the program on your diskette.

**Repeating
Doing the Same Thing Over and Over**

1. Here is a trivial (but fun) example of an infinite loop. Type it in as shown:

```
100 PRINT "JOHN BROWN"  
110 GOTO 100
```

use your own name here

Run the program. The easiest way to stop it is to press CTRL-RESET.

Predict what would happen if you put a semicolon at the end of line 100.

Retype line 100 with a semicolon at the end to try this out.

2. Normally you will want an operation repeated a certain (finite) number of times. Write a new program which uses a FOR NEXT loop to print your name exactly ten times. Use the line numbers 90, 100, and 200 for this program and record your program in the spaces below:

Type: NEW

```
90 _____  
100 _____  
200 _____
```

Run it to see if it works as you expected.

3. Change the program so that it begins by asking the person his or her name once, and then writes that name repeatedly.

Repeating
Doing the Same Thing Over and Over

4. In addition to asking for the name, the program should also ask for:

STREET ADDRESS:

CITY:

STATE:

ZIP CODE:

The program should then print out ten identical mailing addresses in standard form. For instance, if someone typed in information for Charles Washington, the program should print out ten labels of this form:

CHARLES WASHINGTON
210 UNION ST.
NORTHDUCK MN 55117

Print spaces " " between
city, state, and zip.

You now have a useful program. Save it on your diskette.

**Repeating
Using the Counter Number in Graphics**

The counter number in a loop can be printed out, used in calculations, or used for any other function where numbers are used in BASIC. In graphics one could change the COLOR number or the position of a point or line. In this program a series of colored lines will be drawn using a FOR NEXT loop.

Type: NEW

Type in the following lines, then run the resulting program:

```
515 HOME
520 GR
525 REM (* BLUE SKY *)
530 COLOR= 7
540 FOR Y = 0 TO 18
550 HLIN 0,39 AT Y
560 NEXT Y
```

Add these lines, then run the program. again:

```
565 REM (* BLUE WATER *)
570 COLOR= 2
580 FOR Y = 19 TO 39
590 HLIN 0,39 AT Y
600 NEXT Y
```

To make the program more interesting, add a boat:

```
610 REM (* POSITION BOAT *)
620 LET X = 8
705 REM (* A BOAT *)
710 COLOR= 9
730 HLIN X,X + 9 AT 17
740 HLIN X + 1,X + 8 AT 18
745 REM (* A MAST *)
750 COLOR= 13
760 VLIN 9,16 AT X + 4
765 REM (* A SAIL *)
770 COLOR= 15
780 PLOT X + 5,10
790 HLIN X + 5,X + 6 AT 11
800 HLIN X + 5,X + 7 AT 12
810 HLIN X + 5,X + 8 AT 13
```

Notice that line 620 positions the boat. By changing that line, you may place the boat anywhere on the water. Try it, but be careful—the program will stop if any part of the boat goes off the edge of the screen.

SAVE this program with a name like SAILBOAT.

**Repeating
The Bar Graph (Phase 2)**

Now you can change the single bar program you created in Phase 1 so that it accepts input for 12 months:

```

////////////////////
////////////////////
////////////////////

```

TYPE IN A NUMBER (1 - 40):?36

Add FOR and NEXT statements to your graph program so that a number may be typed in and a bar plotted 12 times before the program stops. Use the counter MO for the month number.

One more change must be made to the program. We don't want all of these bars to be plotted in the same place (AT 20). Instead we will plot bars on every third row, starting with row 3 as in the chart below:

month (MO)	row
1	3
2	6
.	.
.	.
12	36

To do this, change your HLIN command to:

HLIN 0,AM-1 AT MO*3

This way, when MO is 1 it will plot on row 3, when MO is 2 it will plot on row 6, and so on.

SAVE this program on your diskette using the name GRAPH 2.

BEGINNING APPLESOFT BASIC

Repeating Exercises III

Using the Counter Number for Animation

1. Animation is the process of linking a series of still pictures together and displaying them in rapid succession to give the illusion of movement. With a computer, images may be rapidly created and displayed. If the progression of the image across the screen is a simple movement, this series of pictures can be created with a loop. The following program is an example of a simple animation of this kind. Type it into the computer and test it out:

```
100 GR
110 FOR Y = 0 TO 39
105 REM (* DRAW A DOT *)
120 COLOR = 7
130 PLOT 20,Y
150 REM (* ERASE IT WITH BLACK *)
160 COLOR = 0
170 PLOT 20,Y
200 NEXT Y
```

After completing this program, save it on your diskette with the name ANIMATE 1.

2. Add the following lines to the ANIMATE 1 program:

```
140 FOR W = 1 TO 20
145 NEXT W
```

Run the program. What do these two additional lines do for the animation?

Why do you suppose this happens?

3. Run and examine the animated version of the SAILBOAT program called MOVING BOAT on your diskette and example program packet. Referring to the diagram and listing in the packet, analyze how this animation is performed.

Making Decisions
The Bar Graph (Phase 3)

or

Aren't We Done with That Thing Yet?

1. Scaling the graph: Load in your most recent bar graph program (Phase 2) which displays 12 months. This program is only useful for graphing values between 1 and 40. To make it more useful you can scale the graph so that each little "block" stands for 5, 10, 100, or any value desired.

First, add two lines at the beginning of the program to ask for and accept the maximum value which would be possible for a bar on this graph (use MX as a variable for this maximum).

Next, add a "scaling calculation" after the line where the amount, AM, is typed in. The scaling calculation should look like this:

```
LET RT = (AM/MX) * 40 - .5
```

RT is the new scaled value for the right endpoint of the bar.

Finally, change the line that plots the bar so that it looks like this:

```
HLIN 0, RT AT MO * 3
```

Now run the program, choosing a maximum of 100 (compatible, for example, with the highest possible score on a test). Try values only in the range of 1 through 100.

SAVE this program as GRAPH 3 on your diskette.

2. Handling small values (forward branch): Rerun the program, using a maximum value of your choice. Run it and type in the first few months in the proper range (up through the maximum).

Type in -300 for one of the months. What happens? _____

There is a problem in using numbers much less than 1/40th of the maximum in this program. In this case, the right endpoint, RT, becomes a negative number. To avoid this problem, make it skip any column in which RT is less than 0. You can use IF THEN to check the input.

Making Decisions
The Bar Graph (Phase 3)

Immediately after the RT calculation line, create a line that looks something like this:

IF RT < 0 THEN 1300 (forward branch)

The underlined line number will probably be different for your program. It is the line number where your NEXT command appears.

Now, if a person using the program types in a zero, a blank space will be left on the graph and the next month's value will be requested. In the space below write down the new IF THEN line you created:

Run the program again and make sure it handles small and negative numbers properly.

3. Handling input errors (backward branch): Now run the graphing program once more. Try typing in a number much greater than the maximum. What happens?
-

This program still needs some work. Take the position that any number greater than the maximum was a typing error. In this case you will want to ask the question again without plotting anything.

Create a new program line after the INPUT statement to check if the number typed in was greater than MX (the maximum). If so, go back to the line where the question was asked, and ask it again.

Write your new IF/THEN line in the space below (backward branch):

Run the program again. This time try typing any number in--large, small and negative. The program should not stop until you have finished month 12.

Save this program as GRAPH 4 on your diskette.

Making Decisions
Multiplication Drill (Phase 1)

1. Type in the following program:

```
100 HOME
110 FOR N = 0 TO 10
120 PRINT N;" X 9 = ";
130 INPUT AN
140 PRINT
300 NEXT N
```

Run the program answering some of the problems correctly and some incorrectly.

What happens when you type a correct answer?

What happens when you type an incorrect answer?

This isn't very satisfactory. Students ought to be told if they get a problem wrong or right. Let's start improving the program. Add the following lines to the program:

```
160 IF AN = N * 9 THEN 300
200 PRINT "NO, THE ANSWER IS ";N*9
290 PRINT
```

LIST the program but don't RUN it yet!

What do you predict will be printed if you get the question wrong?

What do you predict will be printed if you get the question right?

Now RUN the program to try it out.

Making Decisions
Multiplication Drill (Phase 1)

2. The program is better, but it's not very positive when you do well. Now it's time to fix that. Type in the following lines:

```
160 IF AN = N * 9 THEN 230
230 PRINT "RIGHT!"
```

LIST the program but don't RUN it yet!

What do you predict will be printed if you get the question right this time?

What do you predict will be printed if you get the question wrong?

RUN the program again. When you answer a question wrong the response is still unsatisfactory. A well-placed GOTO command will fix it up. Write the GOTO line you need in the space below and add it to your program.

RUN the program now to make sure that it works properly. If it does, SAVE it on your diskette using a name like MATH 1.

3. Extra for experts:

Add an option at the beginning of the program that would ask a question like this:

BY WHAT NUMBER WOULD YOU LIKE TO MULTIPLY?

If someone types 6, the program should test on all the multiples (from 0-10) of 6 (rather than 9 as it does currently).

Finally, at the end of the program ask if they would like to try more problems. If so, go back to the beginning. If not, end the program. SAVE this program on your diskette using a name like MATH 2.

BEGINNING APPLESOFT BASIC

Making Decisions Exercises IV - Page 1

The program BOAT AND ISLAND on your training diskette is an extension of the program MOVING BOAT. There's one problem with it: when the boat passes in front of the island, the island disappears! It is "wiped out" by the boat and the blue background color used in the boat animation. The basic parts of a shipwreck animation sequence are built into this program, if you can only put in the appropriate branches in the right places. A listing of this program may be found in your Reference Packet, page R-9.

1. At the beginning of the program, B (the boat's starting position) and I (the island's position) are set by LET statements. Change this part of the program so that the user can input a starting spot for the boat (1 - 30) and input the position of the island (0 - 31). Be sure to include appropriate prompt lines (PRINT statements).

There are now four different things that could happen to this boat:

IF it is left of the island it should continue sailing.

IF it hits the left of the island it should sink.

IF it is to the right of the island it should continue sailing.

IF it is on the island it shouldn't move at all.

The steps below will allow you to act on each of these four cases. They all fit between lines 880 and 940.

2. Line 890 IF the boat's bow (X+8) hits (=) the island (I), THEN the program should branch to the sinking animation.
3. Line 900 IF the boat's bow (X+8) is to the left (<) of the island (I), THEN the program should branch to the remainder of the animation at line 940.
4. Line 910 IF the boat's stern (X) is to the right (>) of the island's right side (I+8), THEN the program should again branch to the remainder of the animation at line 940.

BEGINNING APPLESOFT BASIC

Making Decisions Exercises IV - Page 2

5. The program would go on to line 940 without any branches at this point. Lines 900 and 910 were added to branch around the following program lines. Type them in:

```
915 REM (* BEACHED CASE *)  
920 PRINT "BEACHED . . . DRAT!"  
925 END
```

This part will keep the boat from moving (END the program) if it is sitting on the island. If it hasn't hit the island, and if it isn't to the left or right of the island, then it must be on the island!

6. If someone tries to start the boat outside of the suggested range (1 - 30), or to place the island outside of its suggested range (0 - 31), the resulting program error will stop the program. With IF THEN statements test to see if each of these answers is in the right range. If not, go back to the point where the question is asked and ask it again. Two IF THEN statements are required after each question to test for both high and low responses.

Accumulators, Counters, and Flags
The Electronic Checkbook Record1. The Accumulator

The following program can help you balance your checkbook. Type it in as shown:

Type: NEW

```
10 HOME
1000 PRINT "STARTING BALANCE ";
1010 INPUT BA
1020 PRINT
1030 PRINT "(- FOR CHECKS, + FOR DEPOSITS)"
1050 PRINT
1060 PRINT "CHECK AMOUNT";
1070 INPUT AM
1080 LET BA = BA + AM
1085 PRINT
1090 PRINT "CURRENT BALANCE ",BA
1300 GOTO 1050
```

Run the program using negative numbers for checks written and positive numbers for deposits to your account. (Use CTRL-RESET to stop the program.)

Line 1080 adds the current balance, BA, and the check or deposit amount, AM. The sum is put into BA as the new balance.

BA is called an "accumulator." It "grows" (accumulates) by adding to itself.

2. The Counter

Add the following lines to your program but do not run the program yet!

```
950 PRINT "LAST CHECK # BALANCED ";
960 INPUT CK
970 HOME
1040 LET CK = CK+1
1060 PRINT "CHECK #";CK
1300 GOTO 1040 ← replace old line 1300
```

Predict: Each time the program goes through the loop this time, what will happen to CK? _____

Run the program to check your prediction.

**Accumulators, Counters, and Flags
The Electronic Checkbook Record**

CK could be considered an accumulator. Since it adds the same number each time, though, it is called a "counter."

3. Is it a deposit?

If you type in a positive number for a check amount, it adds to your account; it is a deposit. Try the program and deposit money several times. What happens to the check number each time you make a deposit?

Since you don't normally use checks to make deposits, your check numbers are all fouled up now.

Add a program line to check whether AM is greater than 0. If so, loop back so that you miss doing line 1040. You must check AM only after the new balance has been calculated and printed out.

In the space below, write the line you added to your program:

Run the program to be sure that checks are not "used up" when a deposit is made.

4. The Flag

Using RESET to stop the program is not very sophisticated. There are several possible ways to gracefully end the program. After each check you could ask whether there were more checks to balance. If there were 50 checks, however, typing "YES" 50 times could become tedious. One alternative is to have a special "code number" to type in as a check amount. When it sees this number the program will stop. For the checkbook application the number 999999 would be good, unless you are likely to make a deposit of that size in the near future!

Add the following lines to your program:

```
1032 PRINT
1033 PRINT "TYPE 999999 TO STOP"
1075 IF AM = 999999 THEN 1500
1500 END
```

Run the program and see how it works.

Save your checkbook program on your diskette.

**Accumulators, Counters, and Flags
More Accumulating and Counting****1. The Bar Graph (one last time!)**

By adding an accumulator to your bar graph program, you could add up all the numbers typed in for the 12 months. Dividing this total by 12 would give the average for the year.

Add the necessary lines to your graphing program to:

- a. total the amounts typed in;
- b. calculate the average;
- c. plot it as a bar in a different color after the bar for month 12.

At the beginning of the program the accumulator should be set to 0.

Print out the average on the screen as the final step.

Save this program as GRAPH 5.

2. Multiplication Drill (Phase 2)

By adding a counter to the multiplication drill you can keep score of the number of right answers. Near the beginning of the program you should set this counter to 0.

The counter should be added to only if the question is answered correctly, so the placement of the "adding line" is critical.

After the last problem, print a message on the screen telling the number right, something like this:

YOU DID 8 OUT OF 11 PROBLEMS CORRECTLY
THAT'S 72.7272727% RIGHT!

The computer can calculate the percentage.

Save this program with the name MATH 3.

Information Concerning the
The following information is being provided to you for your information only.

1. The following information is being provided to you for your information only.

The following information is being provided to you for your information only. It is not to be used for any other purpose.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

2. The following information is being provided to you for your information only.

The following information is being provided to you for your information only. It is not to be used for any other purpose.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

The following information is being provided to you for your information only.

Reference Materials

**BEGINNING APPLESOFT BASIC
Reference Materials Booklet**

Reference

Table of Contents

Apple System Commands for Programmers	R-1
Initializing a Diskette	R-2
Example Programs - Variables	
TAX 1	R-3
TAX 2	R-3
Example Programs - Repeating	
TAX 3	R-4
TAX 4	R-4
VALENTINE	R-5
BOOSHOO	R-6
MOVING BOAT	R-7
Example Programs - Making Decisions	
TAX 5	R-8
BOAT AND ISLAND	R-9
Applesoft BASIC Statements Summary	R-10
Low-Resolution Graphics Planning Sheet	R-14

Apple System Commands for Programmers

Commands Affecting Memory Contents:

NEW	clears the memory of the computer; used to start programming a <u>new</u> program.
RUN	runs the program located in the Apple memory.
LIST	lists the program located in Apple memory. Holding the CONTROL key down and pressing the S key will start and stop the listing.
LIST 530	lists a specific line number (530 in example).
LIST 1000, 1200	lists from the first line listed through the second line (1000 through 1200 in example).

Commands to the Disk Drive:

CATALOG	displays a catalog of programs stored on this diskette.
LOAD programname	copies a program from the diskette into the Apple's memory.
RUN programname	loads a program from the diskette <u>and</u> runs it.
SAVE programname	saves the program which is in the Apple's memory onto the diskette or replaces a previous version on the diskette. You must type a program name after SAVE, not just the word SAVE.
DELETE programname	deletes a program from the diskette.
LOCK programname	restricts a user from saving (replacing) or deleting this program on the diskette.
UNLOCK programname	unlocks a program that has been locked.

Initializing a Diskette

To prepare a new diskette for storing programs, you must first initialize it. Initializing a diskette containing programs will "wipe it clean" of all old programs.

1. With the Apple power turned off, place the DOS 3.3 System Master diskette, which came with your Apple, in the disk drive.
2. Turn on the Apple and wait for the following to appear on the screen.

APPLE II

DOS VERSION 3.3 SYSTEM MASTER

]

3. Take out the System Master and put in the diskette to be initialized.
4. Type in a simple program, such as the following:

```
NEW
10 HOME
20 PRINT "THIS DISK BELONGS TO (your name) "
30 END
INIT HELLO
```

The program you just wrote will be saved as HELLO on the diskette. This program will run automatically when the Apple is turned on with the diskette in the disk drive.

5. Check your diskette by turning off the Apple, placing your new disk in the drive and then turning on the Apple. The screen should clear and the following should appear:

THIS DISK BELONGS TO (your name)

6. Write a diskette label and affix it to the diskette.

TAX 1: Using labels for numbers and words - Variables

```

10 HOME
100 LET NA$ = "NORBERT"
150 LET AM = 15.95
200 LET TX = AM * .04
250 LET TL = AM + TX
300 PRINT "TOTAL PURCHASED BY"
310 PRINT NA$
320 PRINT
350 PRINT "AMOUNT OF PURCHASE:"
360 PRINT AM
370 PRINT
380 PRINT "SALES TAX:"
390 PRINT TX
400 PRINT
410 PRINT "TOTAL:"
420 PRINT TL

```

NA\$	AM	TX	TL

```

10 HOME
90 PRINT "WHAT'S YOUR NAME"
100 INPUT NA$
140 PRINT "WHAT WAS THE AMOUNT OF YOUR PURCHASE"
150 INPUT AM
200 LET TX = AM * .04
250 LET TL = AM + TX
300 PRINT "TOTAL PURCHASED BY"
310 PRINT NA$
320 PRINT
350 PRINT "AMOUNT OF PURCHASE:"
360 PRINT AM
370 PRINT
380 PRINT "SALES TAX:"
390 PRINT TX
400 PRINT
410 PRINT "TOTAL:"
420 PRINT TL

```

R-3

Example Programs - Repeating

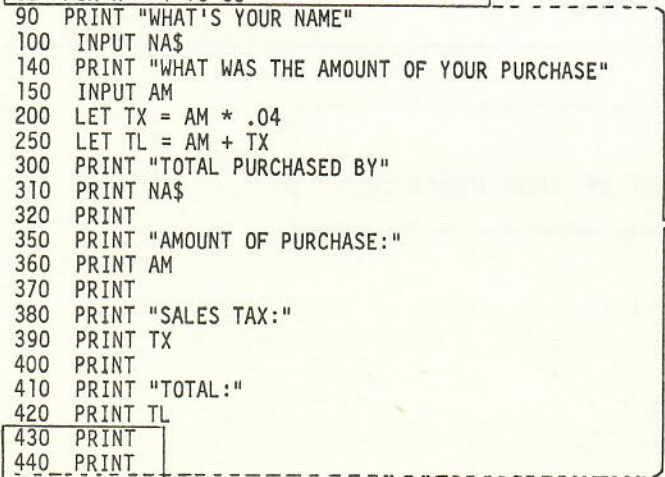
TAX 3: Repeating forever - GOTO

```
10 HOME
90 PRINT "WHAT'S YOUR NAME"
100 INPUT NA$
140 PRINT "WHAT WAS THE AMOUNT OF YOUR PURCHASE"
150 INPUT AM
200 LET TX = AM * .04
250 LET TL = AM + TX
300 PRINT "TOTAL PURCHASED BY"
310 PRINT NA$
320 PRINT
350 PRINT "AMOUNT OF PURCHASE:"
360 PRINT AM
370 PRINT
380 PRINT "SALES TAX:"
390 PRINT TX
400 PRINT
410 PRINT "TOTAL:"
420 PRINT TL
500 PRINT
510 GOTO 90
```



TAX 4: Repeating a certain number of times - FOR NEXT

```
10 HOME
20 PRINT "HOW MANY CUSTOMERS IN LINE"
30 INPUT CU
40 HOME
50 FOR X = 1 TO CU
90 PRINT "WHAT'S YOUR NAME"
100 INPUT NA$
140 PRINT "WHAT WAS THE AMOUNT OF YOUR PURCHASE"
150 INPUT AM
200 LET TX = AM * .04
250 LET TL = AM + TX
300 PRINT "TOTAL PURCHASED BY"
310 PRINT NA$
320 PRINT
350 PRINT "AMOUNT OF PURCHASE:"
360 PRINT AM
370 PRINT
380 PRINT "SALES TAX:"
390 PRINT TX
400 PRINT
410 PRINT "TOTAL:"
420 PRINT TL
430 PRINT
440 PRINT
450 NEXT X
530 END
```



This part is
repeated CU times

Example Programs - Repeating

VALENTINE: Using the counter number to "fill in"

```

1 REM WRITTEN BY SHIRLEY RASMUSSEN
2 REM ROSEVILLE SCHOOLS
3 REM ON FEBRUARY 9, 1983
4 REM AT 7:45 PM
10 HOME
20 GR
30 COLOR= 2
40 VLIN 37,33 AT 8
50 VLIN 37,36 AT 9
60 PLOT 10,37
70 PLOT 10,35
80 PLOT 11,34
90 PLOT 12,33
100 PLOT 13,22
110 COLOR= 13
120 VLIN 4,0 AT 9
130 HLIN 10,12 AT 4
140 HLIN 10,12 AT 0
150 VLIN 4,0 AT 15
160 PLOT 16,4
170 PLOT 17,4
180 VLIN 4,0 AT 18
190 HLIN 20,22 AT 0
200 VLIN 4,1 AT 21
210 VLIN 4,0 AT 25
220 HLIN 26,28 AT 0
230 HLIN 26,27 AT 2
240 HLIN 26,28 AT 4
250 COLOR= 1
260 HLIN 12,14 AT 7
270 HLIN 24,26 AT 7
280 HLIN 11,15 AT 8
290 HLIN 23,27 AT 8
300 HLIN 10,16 AT 9
310 HLIN 22,28 AT 9
320 HLIN 10,17 AT 10
330 HLIN 21,28 AT 10
340 HLIN 9,18 AT 11
350 HLIN 20,29 AT 11
355 FOR X = 12 TO 22
360 HLIN 9,29 AT X
365 NEXT X
370 FOR P = 23 TO 25
380 HLIN 10,28 AT P
385 NEXT P
390 HLIN 11,27 AT 26
400 HLIN 11,27 AT 27
410 HLIN 12,26 AT 28
420 HLIN 12,26 AT 29
430 HLIN 13,25 AT 30
440 HLIN 14,24 AT 31
450 HLIN 15,23 AT 32
460 HLIN 16,22 AT 33
470 HLIN 17,21 AT 34
480 HLIN 18,20 AT 35
490 PLOT 19,36
500 COLOR= 2
510 PLOT 30,9
520 HLIN 31,34 AT 8
530 VLIN 7,5 AT 31
540 PLOT 32,7
550 HLIN 33,36 AT 6
560 VLIN 5,3 AT 33
570 PLOT 34,5
580 PLOT 35,4
590 END
    
```

Arrow Tip

Letters

Heart

CUTE

Shirley Rasmussen

Feather

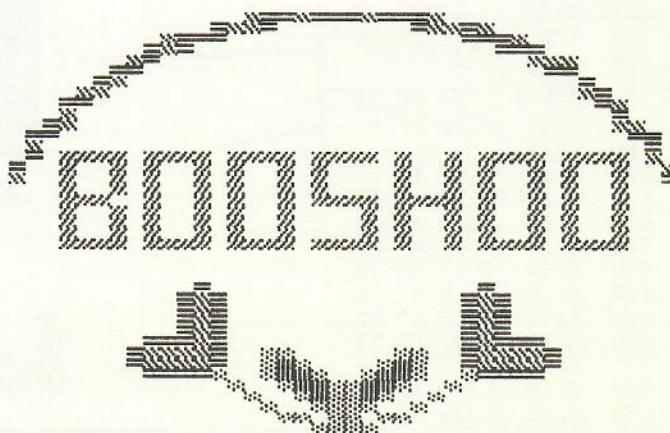
Example Programs - Repeating

BOOSHOO: Using a timing loop for "dramatic effect"

```

1 REM WRITTEN BY CHERIE NEIMA
2 REM RED SCHOOL HOUSE
3 REM ST. PAUL, MN
4 REM BOOSHOO MEANS 'WELCOME' IN OJIBWAY (CHIPPEWA)
5 GR
15 HOME
20 COLOR= 1: REM MAGENTA
25 FOR X = 1 TO 1500
30 NEXT X
40 PLOT 0,16: PLOT 1,13
50 VLIN 10,11 AT 3: HLIN 4,5 AT 8
60 PLOT 6,7: PLOT 7,5
70 VLIN 3,4 AT 9: PLOT 11,3
80 HLIN 13,14 AT 2: PLOT 14,1
90 PLOT 17,1: PLOT 21,1
100 VLIN 1,2 AT 25: PLOT 26,2
110 PLOT 28,2: PLOT 30,3
120 HLIN 31,32 AT 5: PLOT 33,5
130 PLOT 34,8: PLOT 36,9
140 HLIN 36,37 AT 11: PLOT 38,14
145 PLOT 39,15
150 COLOR= 9: REM ORANGE
160 PLOT 0,15: PLOT 1,14
170 PLOT 1,12
180 VLIN 11,12 AT 2: HLIN 3,4 AT 9
190 PLOT 5,7: HLIN 6,7 AT 6
200 VLIN 4,5 AT 8: VLIN 3,4 AT 10
210 PLOT 11,3: VLIN 2,3 AT 12
220 VLIN 1,2 AT 15: PLOT 16,1
230 HLIN 18,20 AT 1: HLIN 22,24 AT 1
240 PLOT 24,2: VLIN 2,3 AT 27
250 HLIN 28,29 AT 3: HLIN 29,31 AT 4
260 HLIN 32,33 AT 6: PLOT 34,7
270 VLIN 8,9 AT 35: PLOT 36,10
280 HLIN 37,38 AT 12: PLOT 38,13
285 PLOT 39,16
290 COLOR= 4: REM DARK GREEN
295 FOR X = 1 TO 1500: NEXT X
300 HLIN 18,21 AT 39: HLIN 16,17 AT 38
305 PLOT 15,37: PLOT 14,36
310 PLOT 13,35: PLOT 12,34
315 HLIN 22,23 AT 38: PLOT 24,37
320 PLOT 25,36: PLOT 26,35
325 PLOT 27,34: VLIN 35,38 AT 19
330 VLIN 35,38 AT 20: PLOT 18,37
335 PLOT 17,36: PLOT 16,35
340 VLIN 32,34 AT 15: PLOT 16,32
345 PLOT 17,33: PLOT 18,34
350 PLOT 21,34: PLOT 22,33
355 PLOT 23,32: VLIN 32,34 AT 24
360 PLOT 23,35: PLOT 22,36
362 PLOT 21,37
365 COLOR= 12: REM GREEN
370 VLIN 33,34 AT 16: VLIN 34,35 AT 17
375 VLIN 35,36 AT 18: VLIN 35,36 AT 21
380 VLIN 34,35 AT 22: VLIN 33,34 AT 23
385 COLOR= 9: REM ORANGE
390 HLIN 8,11 AT 34: HLIN 8,9 AT 31
395 VLIN 32,33 AT 7: VLIN 27,31 AT 10
400 VLIN 27,33 AT 12: PLOT 11,26
405 VLIN 27,33 AT 27: VLIN 27,31 AT 29
410 VLIN 32,33 AT 32: PLOT 28,26
415 HLIN 30,31 AT 31: HLIN 28,31 AT 34
420 COLOR= 1
425 HLIN 8,11 AT 32: HLIN 8,11 AT 33
430 VLIN 27,31 AT 11: VLIN 27,31 AT 28
435 HLIN 28,31 AT 32: HLIN 28,31 AT 33
440 FOR X = 1 TO 1500
445 NEXT X
500 COLOR= 2: REM DARK BLUE
510 VLIN 14,22 AT 3: VLIN 14,17 AT 6
520 VLIN 19,22 AT 6: VLIN 14,22 AT 8
530 VLIN 14,22 AT 11: VLIN 14,22 AT 13
540 VLIN 14,22 AT 16: VLIN 14,18 AT 18
550 VLIN 18,22 AT 21: VLIN 14,22 AT 23
560 VLIN 14,22 AT 26: VLIN 14,22 AT 28
570 VLIN 14,22 AT 31: VLIN 14,22 AT 33
580 VLIN 14,22 AT 36: HLIN 4,5 AT 14
590 HLIN 4,5 AT 18: HLIN 4,5 AT 22
600 HLIN 9,10 AT 14: HLIN 9,10 AT 22
610 HLIN 14,15 AT 14: HLIN 14,15 AT 22
620 HLIN 19,21 AT 14: HLIN 19,20 AT 18
630 HLIN 18,20 AT 22: HLIN 24,25 AT 18
640 HLIN 29,30 AT 14: HLIN 29,30 AT 22
650 HLIN 34,35 AT 14: HLIN 34,35 AT 22

```



Example Programs - Repeating

MOVING BOAT: Using the counter for animation

```

10 TEXT
515 HOME
520 GR
525 REM (* BLUE SKY *)
530 COLOR= 7
540 FOR Y = 0 TO 18
550 HLIN 0,39 AT Y
560 NEXT Y
565 REM (* BLUE WATER *)
570 COLOR= 2
580 FOR Y = 19 TO 39
590 HLIN 0,39 AT Y
600 NEXT Y

```

```

610 REM (* MOVE BOAT *)
630 FOR X = 1 TO 30
705 REM (* A BOAT *)
710 COLOR= 9
730 HLIN X,X + 9 AT 17
740 HLIN X + 1,X + 8 AT 18
745 REM (* A MAST *)
750 COLOR= 13
760 VLIN 9,16 AT X + 4
765 REM (* A SAIL *)
770 COLOR= 15
780 PLOT X + 5,10
790 HLIN X + 5,X + 6 AT 11
800 HLIN X + 5,X + 7 AT 12
810 HLIN X + 5,X + 8 AT 13

```

```

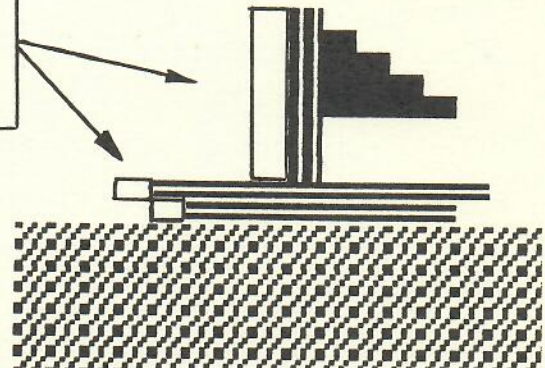
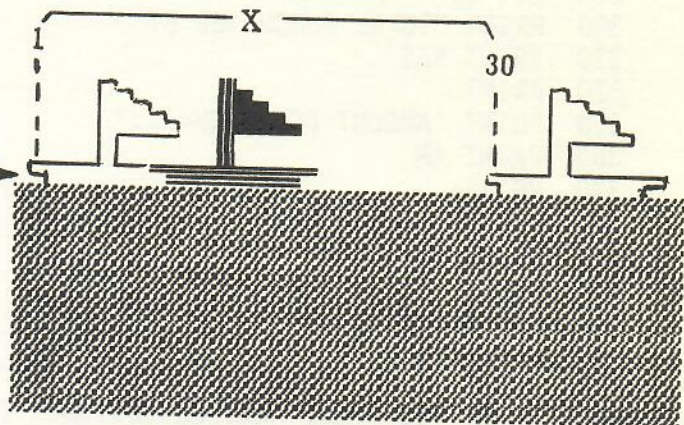
840 REM (* ERASE BACK OF LAST BOAT *)
850 COLOR= 7
860 PLOT X - 1,17
870 PLOT X,18
880 VLIN 9,16 AT X + 3

```

```

940 REM (* SLOW THINGS DOWN *)
950 FOR T = 1 TO 20
960 NEXT T
1000 NEXT X

```



Example Programs - Making Decisions

TAX 5: Making Decisions - IF THEN

```
10 HOME
90 PRINT "WHAT'S YOUR NAME"
100 INPUT NA$
140 PRINT "WHAT WAS THE AMOUNT OF YOUR PURCHASE"
150 INPUT AM
200 LET TX = AM * .04
250 LET TL = AM + TX
300 PRINT "TOTAL PURCHASED BY"
310 PRINT NA$
320 PRINT
350 PRINT "AMOUNT OF PURCHASE:"
360 PRINT AM
370 PRINT
380 PRINT "SALES TAX:"
390 PRINT TX
400 PRINT
410 PRINT "TOTAL:"
420 PRINT TL
470 PRINT
480 PRINT "IS THERE ANOTHER CUSTOMER (Y OR N)"
490 INPUT AN$
510 IF AN$ = "Y" THEN 10
520 HOME
530 END
```

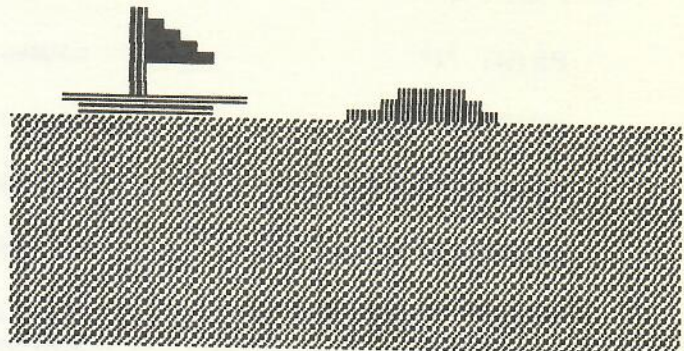

Example Programs - Making Decisions

BOAT AND ISLAND: Just add some IF THENs to make it work!

```

10 TEXT
50 LET B = 1
60 LET I = 20
515 HOME
520 GR
525 REM (* BLUE SKY *)
530 COLOR= 7
540 FOR Y = 0 TO 18
550 HLIN 0,39 AT Y
560 NEXT Y
565 REM (* BLUE WATER *)
570 COLOR= 2
580 FOR Y = 19 TO 39
590 HLIN 0,39 AT Y
600 NEXT Y
601 REM (* GREEN ISLAND *)
602 COLOR= 12
604 HLIN I,I + 8 AT 18
606 HLIN I + 2,I + 7 AT 17
608 HLIN I + 3,I + 6 AT 16
610 REM (* MOVE BOAT *)
630 FOR X = B TO 30
705 REM (* A BOAT *)
710 COLOR= 9
730 HLIN X,X + 9 AT 17
740 HLIN X + 1,X + 8 AT 18
745 REM (* A MAST *)
750 COLOR= 13
760 VLIN 9,16 AT X + 4
765 REM (* A SAIL *)
770 COLOR= 15
780 PLOT X + 5,10
790 HLIN X + 5,X + 6 AT 11
800 HLIN X + 5,X + 7 AT 12
810 HLIN X + 5,X + 8 AT 13
840 REM (* ERASE BACK OF LAST BOAT *)
850 COLOR= 7
860 PLOT X - 1,17
870 PLOT X,18
880 VLIN 9,16 AT X + 3
940 REM (* SLOW THINGS DOWN *)
950 FOR T = 1 TO 20
960 NEXT T
1000 NEXT X
1010 END

```



```

2000 REM (* SINKING ANIMATION *)
2010 COLOR= 7
2020 PLOT X + 8,13
2030 VLIN 12,13 AT X + 7
2040 COLOR= 15
2050 HLIN X + 5,X + 6 AT 14
2060 COLOR= 7
2070 HLIN X,X + 3 AT 17
2080 HLIN X + 5,X + 9 AT 17
2090 VLIN 9,10 AT X + 4
2100 VLIN 10,11 AT X + 5
2110 VLIN 14,11 AT X + 6
2120 HLIN X + 1,X + 8 AT 18
2130 COLOR= 13
2140 VLIN 18,17 AT X + 4
2150 COLOR= 15
2160 VLIN 18,12 AT X + 5
2170 COLOR= 7
2180 FOR Y = 11 TO 18
2190 HLIN X + 4,X + 5 AT Y
2200 FOR T = 1 TO 20
2210 NEXT T
2220 NEXT Y
2290 PRINT "BLUB...BLUB...BLUB!"
2300 END

```

Applesoft BASIC Statements Summary - Output

Printing

PRINT	causes a blank line to be printed
PRINT "HELLO"	causes the word <u>HELLO</u> to be printed
PRINT 7	causes the number <u>7</u> to be printed
PRINT "7"	causes the character <u>7</u> to be printed
PRINT 5 * 8	causes the number <u>40</u> to be printed
PRINT "5 * 8"	causes <u>5 * 8</u> to be printed
PRINT "5 + 7 = "; 5+7	causes <u>5 + 7 = 12</u> to be printed
PRINT "TYPE YOUR NAME ";	using a semicolon (;) at the end of a PRINT statement prevents the computer from automatically going to the next line on the screen.

CalculatingMath Symbols

+ Addition - Subtraction * Multiplication / Division

Order of Mathematical Calculations

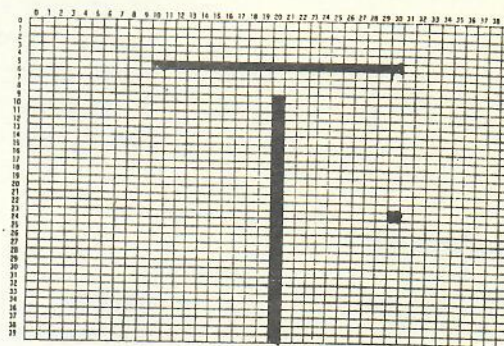
1. Insides of parentheses () are evaluated first
2. * or / next
3. Finally + or -.

Applesoft BASIC Statements Summary - Output

Low-Resolution Graphics

GR

causes the top 20 lines of text on the screen to disappear; this space is now set up for drawing with low-resolution graphics. The low-resolution graphics screen is a 40 by 40 grid of small rectangles, each of which may be plotted in any of 16 different colors.



low-resolution screen grid

COLOR = 12

causes any PLOT, VLIN, or HLIN commands (see below) to draw in the indicated color. Colors include:

0 Black	6 Medium Blue	11 Pink
1 Red	7 Light Blue	12 Green
2 Dark Blue	8 Brown	13 Yellow
3 Purple	9 Orange	14 Aqua
4 Dark Green	10 Gray	15 White
5 Gray		

PLOT 30,23

causes a rectangle to be drawn in the current COLOR value at screen position Horizontal = 30, Vertical = 23. This rectangle is plotted on the sample screen above.

HLIN 10,30 AT 5

causes a horizontal line to be plotted with endpoints at horizontal positions 10 and 30 and positioned down the screen AT 5. This line is plotted above.

VLIN 9,39 AT 20

causes a vertical line to be plotted with endpoints at vertical positions 9 and 39 (bottom) and position across the screen AT 20. This line is plotted above.

TEXT

replaces the graphics screen with a text display.

Applesoft BASIC Statements Summary - Variables**Variables****Numeric variables**

"labels" in which numbers may be stored in a program for later use. A numeric variable label must begin with a letter, and it may be followed by another letter or a number. Examples: N, X, CX, F5.

String variables

"labels" under which any group of characters may be stored in a program. Typically used to store words, sentences, etc. A string variable name must start with a letter and must end with a \$. A second letter or number is also possible. Examples: NA\$, R\$, and S2\$.

LET Statement

LET PI=3.14

causes 3.14 to be stored in the computer's memory with the numeric variable PI.

LET N2\$="2 CARS"

causes the six characters "2 CARS" (includes the space) to be stored for later recall in the string variable N2\$.

INPUT Statement

INPUT NA\$

causes a question mark to be printed and waits for the program user to type something and press RETURN. Whatever is typed will be stored in the string variable NA\$.

INPUT AM

causes a question mark to be printed and waits for the program user to type in a number and press RETURN. Whatever number is typed will be stored in the numeric variable AM. If a non-numeric response is typed in, the program will stop.

Applesoft BASIC Statements Summary - Loops and Branches

Loops

```
550 .....
560 .....
570 .....
580 GOTO 550
```

A GOTO Loop. Each time line 580 is reached, the program sequence begins again at line 550. If there are no commands between lines 550 and 580 which could prevent the program from reaching line 580 (such as an IF THEN branch), the series of commands from 550 through 580 will repeat without limit. It can only be stopped by interrupting the program in some way, such as pressing RESET or turning off the power.

```
130 FOR Q = 1 TO 200
140 .....
150 .....
160 .....
170 NEXT Q
```

A FOR NEXT loop. The program lines from 130 to 170 are executed once with Q having the value 1. At line 170 the program is sent back to start the process over again with the NEXT value of Q (2). The result is that all the lines from 130 through 170 are executed 200 times in this example.

Branches

```
450 GOTO 1230
```

Sometimes called an unconditional branch. When this line is reached, the program must branch to line 1230.

```
110 IF A=5 THEN 500
```

Sometimes called a conditional branch. The program will branch to line 500 only IF a certain condition is true. In this case, the branch will occur only IF variable A is equal to 5.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39

MECC SERVICES

The Minnesota Educational Computing Consortium is an organization established in 1973 to assist Minnesota schools and colleges in implementing educational computing. MECC provides a variety of services to education, including 1) development and distribution of computer software; 2) in-service training for educators and development of materials for conducting training; 3) educational computing assistance through newsletters and computer purchase contracts; and 4) management information services, including the development and maintenance of statewide payroll/personnel and financial accounting software and administrative computer packages. MECC's knowledge and expertise in the educational computing field comes from a decade of working with and providing leadership for hundreds of local educators on a daily basis.

- **MECC Educational Computing Catalog**

Catalogs containing instructional computing courseware, all-purpose training materials, and administrative software are published twice each year and are distributed at no charge. To request a catalog, write or call MECC Distribution (Telephone: 612/481-3527).

- **MECC Memberships**

Non-Minnesota non-profit educational institutions may obtain annual service agreements with MECC which qualify them to obtain MECC courseware and training at special reduced prices. For up-to-date pricing and procedural information on these memberships, write or call MECC Institutional Memberships (Telephone: 612/481-3512).

- **Training Programs**

MECC staff conducts educational computing workshops for educators throughout the United States. For information on workshop schedules, or to arrange a special training activity, write or call MECC User Services (Telephone: 612/638-0626).

- **Administrative Software**

MECC provides a variety of quality administrative microcomputer packages. For information on available packages, training and maintenance contracts, write or call MECC-MIS (Telephone: 612/481-3548).

- **MECC Network Newsletter**

Published five times during the school year, MECC's newsletter focuses on MECC activities, training materials, and educational courseware. To obtain, write or call indicating your interest in the MECC Network Newsletter (Telephone: 612/481-3612).

- **Help Line**

If you have any problems using MECC software with your computer, write or call the Help Line (Telephone: 612/638-0638).

MECC
3490 Lexington Avenue North
St. Paul, MN 55112
(General Information: 612/481-3500)

